AALTO UNIVERSITY
School of Science and Technology

Faculty of Electronics, Communications and Automation

Department of Communications and Networking (Comnet)

Marko Seikola, B. Sc. (Technology)

The Scrum Product Backlog as a Tool for Steering the Product Development in a Large-Scale Organization

Master's thesis for the degree of Master of Science in Technology submitted for inspection.

Espoo, 7th May 2010

Supervisor Timo O. Korhonen, Ph. D. (Technology)

Instructor Kirsi Mikkonen, M. Sc. (Technology)

AALTO UNIVERSITY ABSTRACT OF THE MASTER´S THESIS
School of Science and Technology
Faculty of Electronics, Communications and Automation
Degree program in Telecommunications

Author:	Marko Seikola, B. Sc. (Technology)		
Subject of the thesis:	The Scrum Product Backlog as a Tool for Steering the Product Development in a Large-Scale Organization	Language: English	Number of pages: 100 + 11
Date:	7.5.2010		
Professorship:	User-Oriented Product Development of Telecommunications		
Supervisor:	Timo O. Korhonen, Ph. D. (Technology)		
Instructor:	Kirsi Mikkonen, M. Sc. (Technology)		

The Waterfall model has been widely applied in the software development. However, agile software development methods have emerged to enhance the adaptability to the changing market demands and to utilize evolutionary releases. The most widely adopted agile method is the Scrum framework.

Ericsson Finland is implementing Scrum. To support the transition, the thesis identifies the stakeholders of the product backlog and the data the stakeholders demand and provide. With a simplification, it can be stated that the product backlog is a prioritized list of items to be performed to complete a feature.

Based on a literature review including a benchmark of other telecommunication domain companies, open-ended interviews (n = 6), and semi-structured interviews (n = 11), the thesis lists the stakeholders and their actions at each decision point. The decision framework, which has emerged in-house, was utilized to structure the thesis. A matrix mapping the actions by each stakeholder at each decision point was formed.

The primary data, which the stakeholders require directly and indirectly from the product backlog, are the velocity, work to be done, the date of feature completion, dependencies, costs, and business value. It is important to note that the stakeholders need the data on different levels and for different purposes. Hence, the feasible visualization of the data varies among the stakeholders.

In addition to identifying the stakeholders and the requirements for data, the thesis points out three findings regarding areas to be enhanced. First, multiple stakeholders currently demand their own product backlog. Second, multiple handovers are conducted. Third, the innovation process is isolated.

However, multiple findings supporting the current organizational thinking of the implementation emerged. For instance, the teams are cross-functional, the product ownership responsibility is shared, and the scope of a new feature is optimized to not to include additional functionality that the market does not demand.

Key words: Scrum, product backlog, agile software development methods

AALTO UNIVERSITY
Teknillinen korkeakoulu
Elektroniikan, tietoliikenteen ja automaation tiedekunta
Tietoliikennetekniikan tutkinto-ohjelma

DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	TkK Marko Seikola		
Työn nimi:	The Scrum Product Backlog as a Tool for Steering the Product Development in a Large-Scale Organization	Kieli:	Sivumäärä:
Päivämäärä:	7.5.2010	Englanti	100 + 11
Tutkinto-ohjelma:	Tietoliikennetekniikan käyttäjäkeskeinen tuotekehitys		
Valvoja:	Dos. Timo O. Korhonen		
Ohjaaja:	DI Kirsi Mikkonen		

Vesiputousmalli ja sen variaatiot ovat olleet laajalti käytössä ohjelmistotuotannossa. Näiden mallien vikojen korjaamiseksi, eli markkinoiden vaatimuksiin mukautumisen kohentamiseksi sekä evolutionististen toimitusten tekemiseksi, on syntynyt ketteriä ohjelmistokehitysmenetelmiä. Näistä eniten käytetty on scrum-viitekehys.

Suomen Ericsson on ottamassa scrum-menetelmän käyttöön. Muutoksen tukemiseksi, tämä diplomityö tunnistaa product backlog -tehtävälistan asianosaisia sekä heidän tarvitsemaansa ja tuottamaansa tietoa.

Kirjallisuuskatsauksen, tietoliikennealan yritysten vertailun, strukturoimattomien haastatteluiden (n = 6) sekä puolistrukturoitujen haastatteluiden (n = 11) avulla tutkimuksessa tunnistettiin asianosaiset ja heidän toimintansa eri päätöspisteissä. Pohjana tutkimuksessa käytetään yrityksen uutta päätöksentekoviitekehystä. Diplomityö esittelee taulukon avulla asianosaisten toimenpiteet eri päätöspisteissä nimenomaan product backlog -tehtävälistaan liittyen.

On oleellista huomata, että eri asianomaiset tarvitsevat eri tietoa. Lisäksi eri vastuuhenkilöille sopivat erilaiset visualisointitavat. Näin ollen, työ esittelee myös viitekehyksen, jonka avulla visualisointia voidaan kohdentaa eri asianomaisille.

Tutkimuksessa havaittiin kolme seikkaa, joita tulee kohentaa. Ensinnäkin, useat asianomistajat haluavat oman backlog -tehtävälistan. Toiseksi, tehtäviä ja vastuita siirretään useassa kohtaa toiselle henkilölle. Kolmanneksi, innovaatioprosessi on erillään uuden tuotteen kehittämisestä.

Toisaalta, useita kaavailtua tapaa tukevia löydöksiä havaittiin kirjallisuudesta. Esimerkiksi, yrityksen tiimit ovat monialaisia, tuotevastuu on hajautettu useammalle henkilölle sekä uuden ominaisuuden sisältö on rajattu vastaamaan markkinan todellista vaatimusta.

Avainsanat: Scrum, product backlog -tehtävälista, ketterät ohjelmistotuotantomenetelmät

Acknowledgements

I have received support from many directions for my thesis. First of all, I would like to thank my employer, Ericsson Finland, for providing me the opportunity to write the thesis regarding an interesting and hot topic.

I thank my instructor Kirsi Mikkonen (M. Sc. Tech.) and the co-instructors Risto Kivoja (M. Sc. Tech.) and Juha Eloranta (M. Sc.). I am grateful for the support, valuable comments, and encouraging I have received from you over the last four months. No matter what was the problem, you were always there to help.

I want to thank my supervisor Timo Korhonen (Ph. D., Tech.) not only for the support regarding my thesis but also for the support for my early career and learning. I am grateful for the cooperation we have had first with my Bachelor's Thesis and now with my Master's Thesis.

I would also like to thank all the interviewees. Without the knowledge I gained from you, the thesis would not exist. You all had a very positive attitude towards the interviews and my thesis. I am also grateful for my colleagues Mikko Pesonen (M. Sc. Tech.) and Aila Koponen (M. Sc.) regarding the comments you provided me after reading my thesis.

Last but not least, I want to thank my girlfriend Maija, my parents Aila and Harri, and my friends for encouraging me with the thesis.

Jorvas, Kirkkonummi, 7.5.2010

Marko Seikola

Table of Contents

Index of Figures

Index of Tables

Abbreviations

2G	Second generation mobile network, i.e., GSM
3G	Third generation mobile network
AgileEVM	Agile Earned Value Management
APO	Area Product Owner
ASD	Adaptive Software Development
BV	Business Value
BVI	Business Value Index
CPI	Cost Performance Index
DSDM	Dynamic Systems Development Method
EBV	Earned Business Value
EPP	Early Phase Program
FCS	Feature Concept Study
FDD	Feature Driven Development
GA	General Availability
GSM	Global System for Mobile Communications
IRR	Internal Rate of Return
JIT	Just-In-Time
M-MGw	Ericsson Mobile Media Gateway
MMF	Minimum Marketable Feature
MSS	Ericsson Mobile Softswitch
MTI	Management of Technological Innovation
NPD	New Product Development
NPV	Net Present Value
PB	Product Backlog
PO	Product Owner
PPO	Proxy Product Owner
PP	Pair Programming
R&D	Research and Development
ROI	Return on Investment
ROIF	Return on Investment Factor
RUP	Rational Unified Process
SM	Scrum Master
SP	Story Point
SPI	Schedule Performance Index
TDD	Test Driven Development
TR	Trouble Report
XFT	Cross-Functional Team
XP	Extreme Programming

Key Concepts

Agile

The agile software development models enhance the adaptability to the customer demands. Different tasks, e.g., coding and testing, are performed simultaneously. In addition, waste is minimized. The product is delivered with an evolutionary approach, i.e., the software is released incrementally. (Deemer et al., 2008; Nerur, 2005; Schwaber, 2007; Sutherland, 2004)

Large-Scale Implementation and Scaling

Large-scale implementation refers to utilizing agile throughout the organization. The transition from an agile pilot to the enterprise level agile implementation is called scaling. (Larman & Vodde, 2009)

Scrum

Scrum is a software development framework that emphasizes the cross-functional teams. The teams perform the tasks based on the items in the product backlog. The term for iteration in Scrum is 'sprint'. A sprint may last from one to four weeks. At the end of each sprint, a potentially shippable product increment is completed. (Deemer et al., 2008; Schwaber, 2009)

The Product Backlog

The product backlog (PB) includes a prioritized list of all features to be implemented. The features are split into multiple items, which are written in form of user stories. The product owner is responsible for the product backlog. (Schwaber 2007, 2009)

Velocity

The velocity refers to the number of completed product backlog items. The velocity data is utilized in estimating the date of feature completion. Thus, it is a tool for managing the release scope. In addition, the scrum teams utilize the concept of velocity for planning the next sprint. (Kniber, 2007)

The Decision Framework

Over the scrum pilot in the case company, an in-house decision framework for the new feature development had been developed prior to the thesis. The framework consists of five decision points, starting from 'F0' and ending to 'F4'. The stages F0 and F1 involve the initial studies of a new feature, i.e., a one-page document is conducted. At F2, the investment decision is formed. At F3, the teams commit to the feature release date. Finally, at F4, the feature is ready, denoting that the coding, the function testing, and the early system testing are executed.

The State-Gate Model

Cooper (2000, 2001) developed the state-gate model for managing the new product development. The model includes five gates and five states. The stages are for executing the development work. Each stage ends in a gate. The gates are for forming decisions regarding the future actions.

1 Introduction

1.1 Motivation

Products and services are outcomes of development projects. A project involves an ideation phase, an execution phase, and a support phase. Most of the decisions are conducted over the execution phase. The closer to the product release the decision is formed, the less it involves risk. (Artto, 2010) Thus, the predefined decisions form the major risks. Figure 1 illustrates the importance and risk of decisions as a function of the project state.

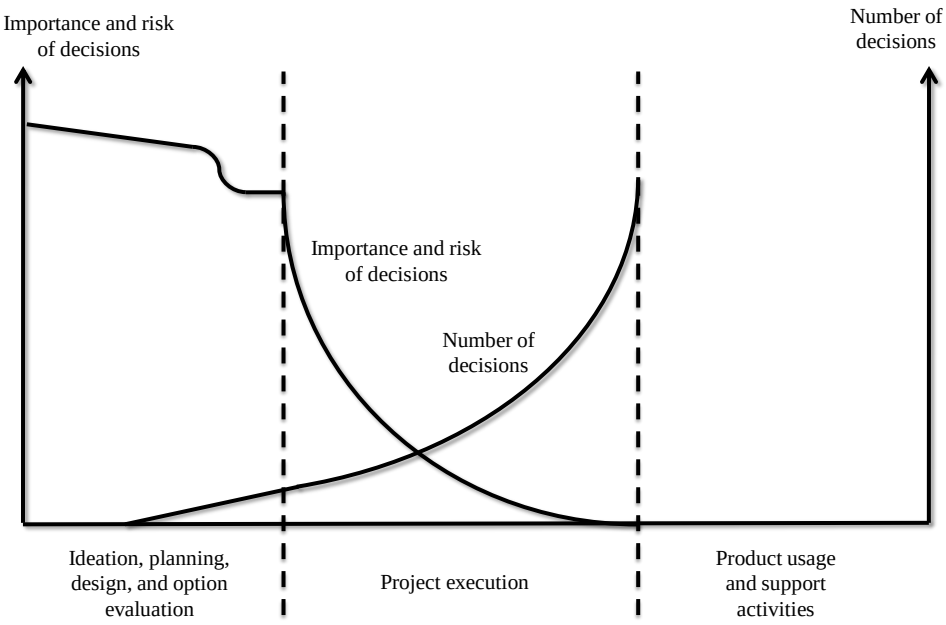

Figure 1. Importance and risk of decisions (Artto, 2010)

In the software development domain, one of the earliest software development process models was the Waterfall model. However, already the author of the model, Winston Royce (1970), identified the risk of long feedback loops. The model has been criticized because of the lack of flexibility and the incentive to predefine the product. Since the inflexibility leaded to long lead times and wasted effort, the trend has been to discard the Waterfall model and to implement the agile methods. (Deemer et al., 2008; Schwaber, 2007; Sutherland, 2004)

The agile software development methods emphasize flexibility and adaptability. The product is delivered to the markets incrementally. Thus, the lead time for a single feature can be decreased with the help of agile meanwhile reducing the length of the feedback loops. (Deemer et al., 2008; Schwaber, 2007; Sutherland, 2004) Scrum is the most widely adopted agile framework (Koskela, 2009).

In the telecommunications domain, the current trends are for instance interoperability, mobility, services, and integration with the internet. The boundaries are formed by, e.g., the regulations and the available spectrums for radio communication. (Ericsson, 2009)

The domain involves embedded software development projects, i.e., the software is bundled to the hardware. The ratio of the amount of software compared with the amount of hardware has increased meanwhile the products have become increasingly complex.

The market shares among the network equipment vendors have changed since some of the traditional players have declined over the economic distress. For instance, the Canadian company Nortel filed for bankruptcy in 2009 (Wahl, 2009). The agile software development methods support companies in adapting to the changes in the market demands (Deemer et al., 2008; Schwaber, 2007; Sutherland, 2004).

Ericsson Finland R&D develops the Mobile Media Gateway (M-MGw). The gateway is a node in the 2G and 3G mobile networks. It connects different transition protocols. The product is partly modular and involves platforms. Hence, the new features may impact multiple existing modules. The product includes complicated dependencies that are not self-evident. As the other products in the telecommunications domain, the Mobile Media Gateway is an embedded software product, which leads to an even increased number of dependencies and increased complexity.

The Mobile Media Gateway is part of Ericsson Mobile Softswitch (MSS) product portfolio. Frequently, the customer requests are routed via the MSS-level since the changes might affect multiple products. Thus, the gateway has internal dependencies to other MSS products but also external dependencies to the third parties. The number of engineers involved in the gateway product development is large. In addition to Finland, also engineers in other Ericsson sites, for example in Hungary, are involved in the development of the product.

Since the market is changing, Ericsson targets at releasing increments to the Mobile Media Gateway in smaller portions than earlier. The feedback loops are shortened and the amount of wasted software is reduced. The tool for achieving these targets is the Scrum framework. The framework emphasizes self-organization, cross-functionality, the short cycles of development, the integration of the different phases of the development, and collaboration. Hence, the efficiency and collaboration are enhanced. The requirements for the new features are presented in an artifact called ‘product backlog’. (Deemer et al., 2008; Kniber, 2007; Schwaber, 2009) However, Scrum is only a framework – it does not state the engineering practices (Schwaber, 2009). The practices can be managed for example with Extreme Programming (XP) (Kniber, 2007; Sutherland, 2001).

At the beginning of the research, the Scrum framework was experimented in the gateway development with two teams in Finland and one team in Hungary. The vision was to scale Scrum to the large extent. According to literature (for instance Leffingwell, 2007; Lyon & Evans, 2008; Kalliney, 2009), the scaling of the Scrum framework is challenging. The challenges vary between the companies. One of the major challenges has been the distribution of the development into multiple sites (Paasivaara et al., 2008, 2009).

Since the scaling of the Scrum framework is a demanding task, there was a need for a study in the context of Ericsson Finland. The scope of the study is the product backlog and the data it provides directly and indirectly, since, according to the literature review, multiple options for implementing the backlog in a large-scale product development exist (Chapter 2.4.1 Options for Scaling).

1.2 Research Problem

The product backlog includes requirements for the features to be implemented. The backlog includes also an estimate for the effort needed to complete a feature and an estimate for the business value (BV). Based on these figures, the value and the date of feature completion can be estimated. (Deemer et al., 2008; Schwaber 2009) Hence, the product backlog is an important element in managing the product development process. Based on that, the high-level research framework was stated to be

How to utilize the scrum product backlog as a tool for steering development work in a large scale organization?

Since the question is too wide to be answered, two sub-questions were introduced. The thesis does not define a solution to the high-level question directly but defines a solution to two focused questions.

It is obvious that the scrum roles, i.e., the team, the scrum master, and the product owner, utilize the product backlog (Schwaber, 2009; Kniber, 2007) but the product backlog might involve potential non-obvious users. For instance, the people responsible for the release management exploit the product backlog as a tool in their work. In addition, the product backlog might provide new insights and interconnections to the innovation process. The different stakeholders interpret the product backlog in different decision points. Thus, they may provide specific requirements for the product backlog implementation. Hence, the first of two sub-questions was stated as follows:

Q1: Who are the stakeholders of the product backlog and how do they exploit it?

The stakeholders of the product backlog (later stakeholders) are interested in different outputs, i.e., in the different data the product backlog provides directly and indirectly. Therefore, the second sub-question was stated as follows:

Q2: Which product backlog data is utilized in steering the product development and how?

The second sub-question addresses on identifying direct and indirect data the product backlog offers and the relevance of the data for different stakeholders. Thus, the second sub-question tights the first sub-question to the implementation of the product backlog. The scope of the thesis is to study the development of the new features. The thesis does not focus in a comprehensive manner, e.g., on the trouble report (TR) handling although the trouble reports can also be managed via the product backlog.

1.3 Research Methods

First, literature regarding the scaling of the Scrum framework was examined. In this thesis, the scaling refers to the transition from the trial teams to the large-scale implementation of Scrum in a distributed product development. Also, a benchmark was conducted, i.e., literature regarding scaling the agile methods in the telecommunications domain was studied.

In addition, literature was reviewed to create a knowledge base of evolution of software development models as well as of the Scrum framework. The thesis relies on academic papers as well as on presentations of the scrum conferences. In addition, the thesis studies publications and presentations regarding agile by companies operating in the telecommunications domain.

To define a solution for the first question, a matrix was created. The matrix includes the stakeholders of the product backlog mapped to the decision points. The matrix was formed based on the interviews with the stakeholders, and on the conclusions drawn in the thesis. The matrix was reflected on literature to identify similarities and differences.

In total, six open-ended interviews formed the basis for determining the research questions and provided knowledge regarding the decision framework. In addition, the stakeholders were identified based on the open-ended interviews. The data for the matrix was gathered in eleven semi-structured interviews. A blank template of the matrix already described was involved as a framework for these interviews.

The matrix and the conclusions based on it form the solution for the second question. In addition, suggestions for the means of visualizing the data to meet the expectations of different stakeholders were discussed.

1.4 Research Outputs

The thesis adds value to the case company by providing insights into the product backlog, mapped to the decision framework. The insights are presented in form of a matrix. Based on the created matrix, dependencies can be identified. In addition, reflections from the literature are stated. Based on the literature review, both positive and negative findings were identified.

The thesis identifies the most relevant data provided directly and indirectly by the product backlog. The data is discussed stakeholder by stakeholder including the suggestions of means of visualizing the data. Hence, the thesis provides a tentative framework for managing the direct and indirect data by the product backlog.

1.5 Structure of the Thesis

Figure 2 visualizes the structure of the thesis. Next in Chapter 2, literature is examined. The first step is to discuss the evolution of software development models. The historical perspective begins from the introduction of the Waterfall model in 1970. It is important

to understand the basics of evolution of software development models in order to interpret the feasibility of agility. The description of agile and lean will follow along with the description of the Scrum framework. First the framework is discussed in overall followed by the introduction of each role, artifact, and meeting related to Scrum. In addition, the concepts of the velocity and the definition of done are discussed. The literature review is concluded by insights from literature regarding scaling the agile methods and benchmarking the telecommunications domain.

Research methods are presented in Chapter 3. The study begins in Chapter 4 by introducing the decision framework developed by the in-house agile experts. Next, each stakeholder is discussed one by one. Chapter 5 focuses on offering insights to the means the different stakeholders utilize the direct and indirect product backlog data. In addition, Chapter 5 presents options for visualizing the data.

Chapter 6 summarizes the study by interpreting the conclusions from the interviews mapped to the literature review. Finally, chapter 7 is devoted to the discussion. Appendix D illustrates the matrix combining the stakeholders and the decision points.

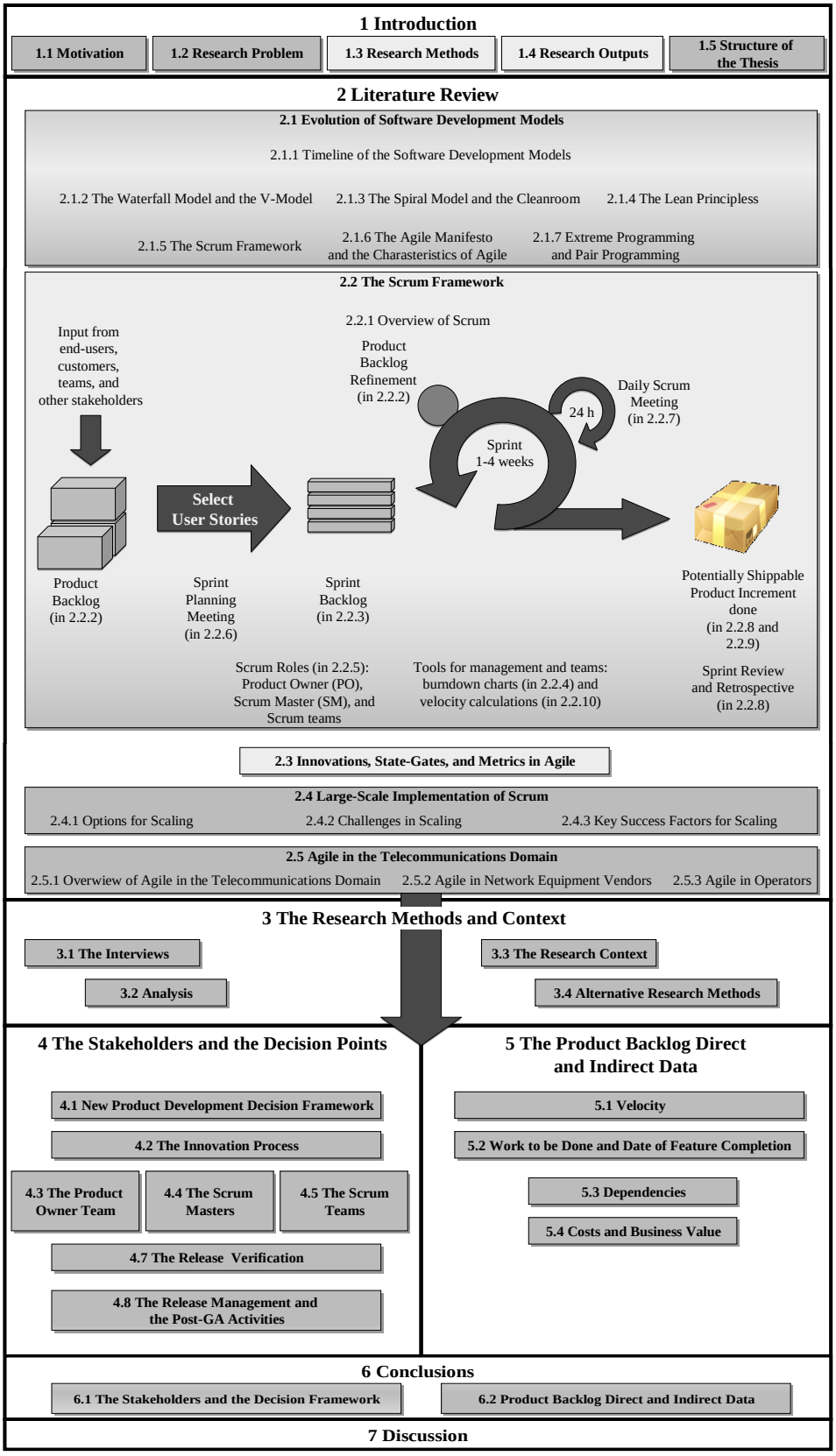

Figure 2. The structure of the thesis

2 Literature Review

The software development models have existed only for a short period of time. The chapter begins by introducing the path from the development of the Waterfall model to the development of the Scrum framework. Along the path, the concepts of lean and agile are explained in addition to a short description of relevant software development methods introduced in the past. The Scrum framework is described deeply, i.e., each role, artifact, and meeting is described. In addition, the chapter defines means of scaling Scrum in large and complex organizations. Also, the agile methods in the telecommunications domain are discussed.

2.1 Evolution of Software Development Models

2.1.1 Timeline of the Software Development Models

One of the first defined models for software development was the Waterfall in 1970. However, since the major disadvantages of the model, new models emerged. Through the development of Cleanroom and the Spiral model, software development models were emerging towards lean and agile. (Abrahamsson et al., 2003) Figure 3 places the cornerstones of software development models on a timeline. Next, the highlighted methods are examined.

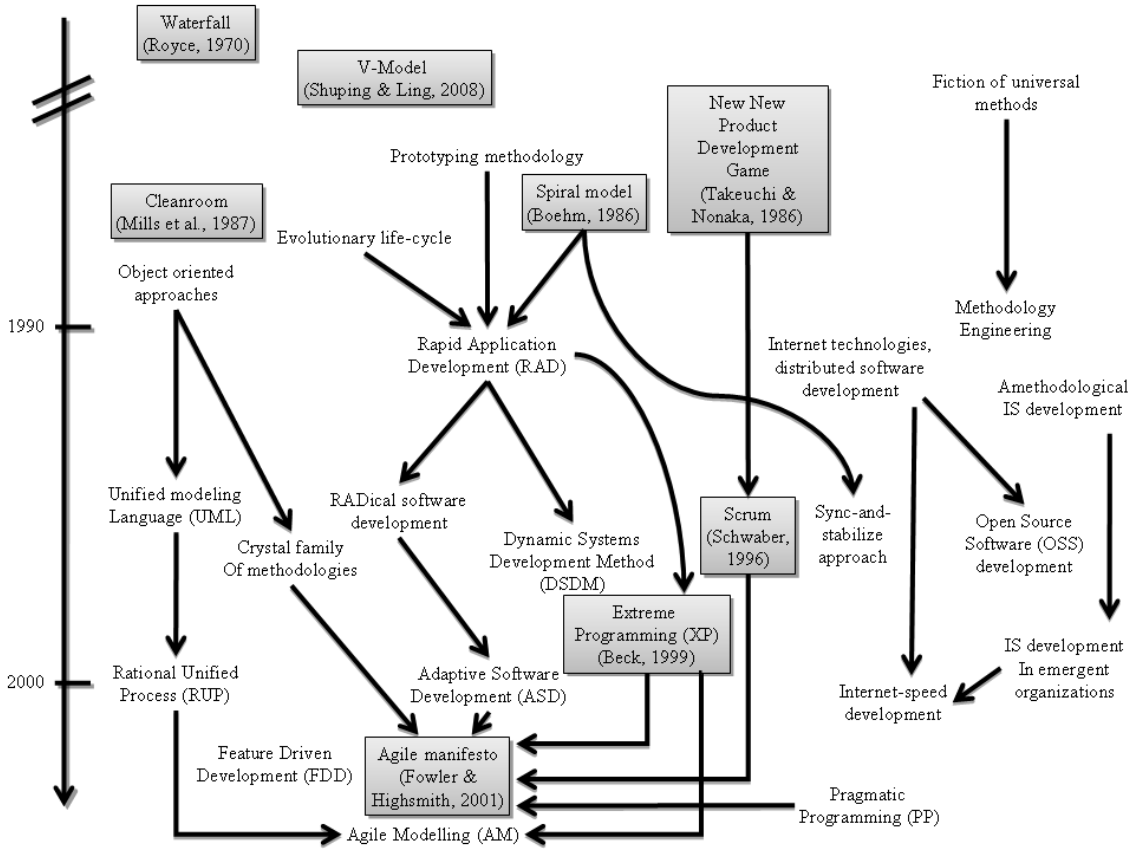

Figure 3. The framework for Chapter 2.1 (modified from Abrahamsson et al., 2003). Note the stated main sources of the methods to be discussed.

2.1.2 The Waterfall Model and the V-Model

The Waterfall model was first introduced by Winston Royce (1970). However, Royce did not yet involve the term 'Waterfall' – he involved the term 'managing the development of large systems'. In the initial paper, Royce identified that there are risks in the implementation of the model and thus it can easily be misused. Figure 4 illustrates the original model from the year 1970. The process is sequential and emphasizes the importance of planning. There are four phases of planning prior to the coding phase as follows: system requirements, software requirements, analysis, and program design. The testing is executed in the second last phase. (Royce, 1970)

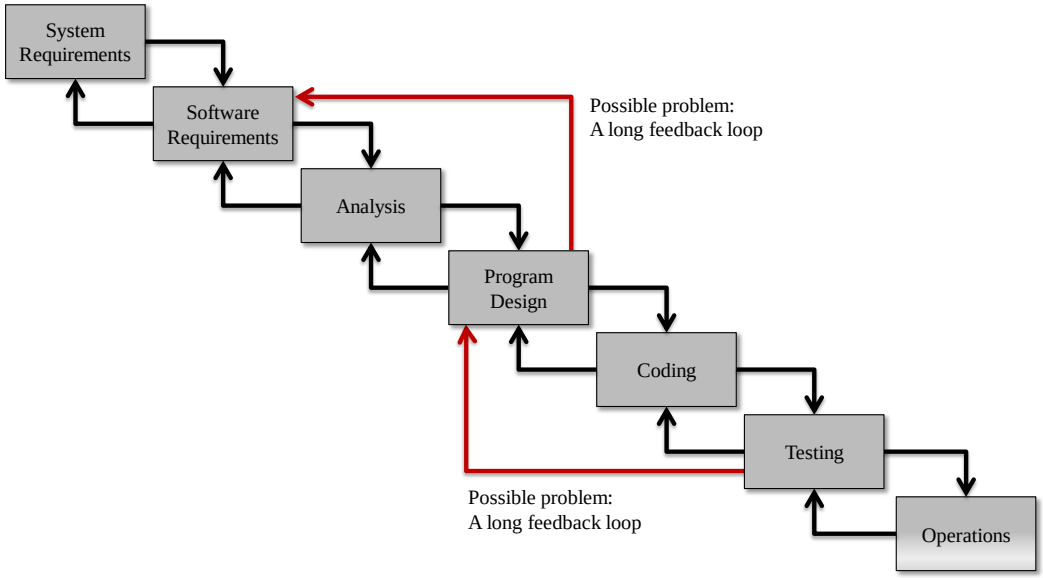

Figure 4. The original Waterfall model (modified from Royce, 1970)

Royce (1970) identified the major challenge to be the faulty implementation of the model. This implementation might lead to the fact that a fault identified in the testing phase is reported to the program design and even further to the software requirement definition. Thus, the faults identified late set at least part of the process back on the origins. In Figure 4, possible problems are illustrated with the red arrows. (Royce, 1970)

The common features of the Waterfall model, and its modifications, are the centralized teams performing functionalities, strict control, and follow-up with Gantt charts. Since the process is strictly following the phases presented in Figure 4, all of the new concepts generated over the process are expensive and time-consuming to implement since the process needs to go back to the design and even further to the requirement specification. Hence, adapting to the changing customer needs is inflexible. (Deemer et al., 2008) Overall, Royce originally identified the issues that later became the major obstacles to the Waterfall model (Deemer et al., 2008; Royce, 1970).

The V-model shares multiple common characteristics with the Waterfall model. The model is named after the letter 'V' since the procedure follows a downward slope and then an upward slope. The underlying concept is to specify requirements, design, and

code on the downward slope, i.e., the left hand side of the letter 'V', and perform unit testing, integration testing, system testing and acceptance testing on the upward slope, i.e., the right hand side of the letter 'V'. As in the Waterfall model, the errors are identified late in the project utilizing the V-model. (Shuping & Ling, 2008)

2.1.3 The Spiral Model and the Cleanroom

Boehm (1986) tried to establish a more efficient and iterative model than the Waterfall. He introduced a risk-driven approach called 'the Spiral model'. The Spiral model begins by defining the product life cycle. Risk analysis and a prototype are created at each cycle of the spiral. The last prototype is operational. Each of the cycles begins by identifying the objectives and the alternative means of implementing the product. However, if there is only a low technological risk, the model will be similar to the Waterfall since the tasks are performed sequentially. (Boehm, 1986) Figure 5 visualizes the spiral model.

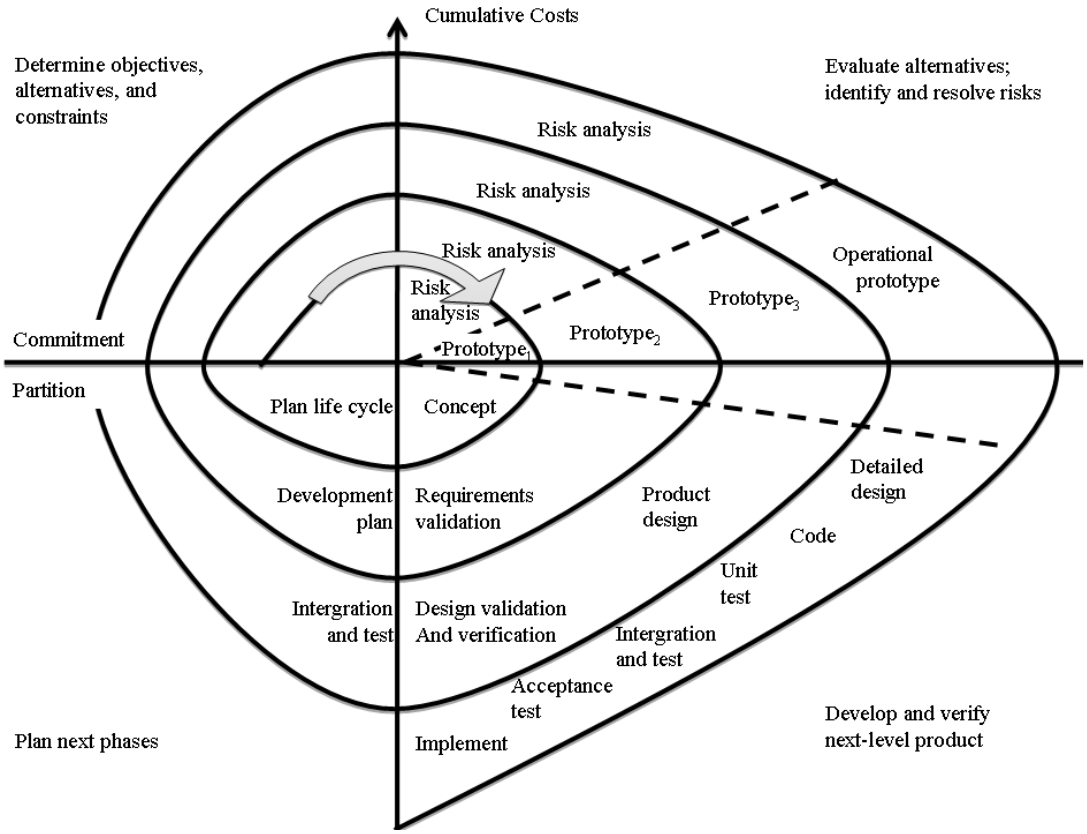

Figure 5. The Spiral model (Boehm, 1986)

Four phases for each cycle can be identified in the Spiral model. In the first phase the objectives, the limitations, and the alternatives are determined. In the second phase, the risks and the alternatives are evaluated. The actual development, including design, the specifications, coding, and testing, is performed in the third phase. The fourth phase is for planning the future activities and for evaluating the cycle. (Guimarães and Vilela, 2005)

Over the same era, the model called ‘Cleanroom’ was developed at IBM. The model introduced incremental product development. However, the statement was that the requirements are to be defined early and not changed over the development. The model emphasized formality and statistical quality control. The underlying concept was to measure quality mathematically. (Mills, 1987)

2.1.4 The Lean Principles

The lean principles originate from the car manufacturing process at Toyota. The underlying concept is to provide the wanted product or service to the customer at the wanted place at the wanted time. The waste is simultaneously minimized. The lean thinking is related also to Just-In-Time (JIT). (Poppendieck, 2007; Raman, 1998) The lean thinking can be utilized as a basis for understanding the agile methods (Poppendieck, 2007).

Raman (1998) presents five lean principles in the context of embedded software development; value, value stream, flow, pull, and perfection. The value can be experienced only by the customer. The value stream, i.e., the combination of actions to produce the product or service, is identified in order to eliminate the ones that do not add value. Flow refers to the coordination and continuous action. The sub-optimizing is not allowed. Pull refers to delivering only the desired features to the customer. The last principle, perfection, includes the continuous improvement and elimination of both the waste and the defects. (Raman, 1998) As a complement, Benefield (2008) identified seven wastes of software development; extra features, partially done work, extra effort, handoffs, task switching, delays, and defects. Poppendieck (2002) states the seven wastes to be as follows: extra features, requirements, extra steps, finding information, defects not caught by tests, waiting, and handoffs.

The principles by Raman (1998) and waste definitions by Benefield (2008) and Poppendieck (2002) are supporting each other. In addition, Poppendieck (2007) presents principles including for instance, quality, knowledge creation, optimizing the whole, delivering rapidly, and minimizing waste. She also argued that lean increase the quality and speed.

2.1.5 The Scrum Framework

Harvard Business Review Article “The New New Product Development Game” (Takeuchi and Nonaka, 1986) was the first paper discussing Scrum. The study examined the new product development in such multinational companies as Honda, Hewlett-Packard, 3M, and Xerox. The paper suggests a framework, ‘Scrum’, which included six characteristics; built-in instability, self-organizing project teams, overlapping development phases, multi-learning, subtle control, and organizational transfer of learning. In addition, the article emphasize the differences between sequential, partly overlapping, and the highly overlapping phases of development. The term ‘Scrum’ originates from the sports rugby. (Takeuchi & Nonaka, 1986) In rugby, Scrum refers to restarting the game after an infraction (Brown et al., 2007, pp. 20).

In the early 1990's Ken Schwaber was developing the Scrum framework at Advanced Development Methods Inc. He realized that creating software is an empirical process. Thus, he emphasized adaptability, partitioning, and the iterative incremental approach instead of a highly predefined product development. He argued that Scrum would enable a major increase in productivity. (Advanced Development Methods, 1996)

In 1993, Jeff Sutherland introduced the first Scrum trial at the Easel Corporation. His motto was "all at once", referring to simultaneous analysis, design, coding, and testing. He discarded Gantt charts and introduced the transparent 30-day sprints of product development. The outcomes from the first trial of Scrum were promising. Delivery times were met, the progress was predictable, and inflexibility adjusting to the changing customer needs was solved. (Sutherland, 2004) The first large organization to experiment Scrum was a healthcare software company IDX in the United States. The company employed hundreds of people. The experiment proved that the Scrum framework can scale to a large organization. However, there was variance in the productivity between the teams. Some teams entered a hyper-productive state whereas others did not. (Sutherland, 2001)

Ken Schwaber (1996) presented the Scrum framework for the software development context with Jeff Sutherland in the OOPLSA seminar in 1995. The framework emphasized the flexibility to respond to sudden changes. The key characteristics were similar to the ones Takeuchi and Nonaka (1986) presented. The initial Scrum framework by Schwaber and Sutherland stated that the Scrum methodology consists of three phases; pre-game, game, and post-game. The pre-game included the planning and high-level designing. The game itself included the sprints and the reviews. The post-game included a closure. (Schwaber, 1996)

2.1.6 The Agile Manifesto and the Characteristics of Agile

In 2001, post the introduction of Scrum (Schwaber, 1996), the Agile Manifesto was written. A group of 17 software development experts, including Schwaber and Sutherland, met at a ski resort. They agreed on twelve principles for agile software development. Appendix A lists the principles. In addition, they presented four values; the individuals and interactions over the processes and tools, the working software over the comprehensive documentation, the customer collaboration over the contract negotiation, and responding to change over following a plan. The guideline was named as 'the Agile Manifesto'. (Fowler & Highsmith, 2001)

The traditional software development models lack flexibility, i.e., the projects cannot be dynamically adjusted. The agile methods address to solve the problem. (Deemer et al., 2008; Nerur, 2005; Schwaber, 2007; Sutherland, 2004) According to Cohn (2007), the customer involvement remains steady throughout the project with the agile methods, whereas with the traditional methods, the customer is mostly involved only at the beginning and at the end of the project. Table 1 summarizes the major characteristics of both the traditional and the agile software development methods.

Table 1. Comparing the traditional and agile development models (Nerur, 2005)

	Traditional	**Agile**
Fundamental Assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning	High-quality, adaptive software can be developed by small teams using the principles of continuous design improvement and testing based on rapid feedback and change
Control	Process-centric	People-centric
Management Style	Command-and-control	Leadership-and-collaboration
Knowledge Management	Explicit	Tacit
Role Assignment	Individual, favors specialization	Self-organizing teams, encourages role interchangeability
Communication	Formal	Informal
Customer Role	Important	Critical
Project Cycle	Guided by tasks and activities	Guided by product features
Development Model	Life cycle model (Waterfall, Spiral, or some variation)	The evolutionary delivery model
Desired Organizational Structure	Mechanistic (bureaucratic with high formalization)	Organic (flexible and participative encouraging cooperative social action)
Technology	No restriction	Favors object-oriented technology

According to a study involving 150 French companies (The French Scrum User Group, 2009), the major reasons for adopting the agile software development models were ability to integrate change (78%), quality improvement (62%), enhancing motivation (61%), and delivering more frequent (49%). The other reasons related to meeting deadlines, enhancing time-to-market, reducing risk, and increasing cost-efficiency. Among others, the survey involved telecom domain companies such as Alcatel-Lucent and France Telecom. Most of the involved companies were utilizing Scrum. (The French Scrum User Group, 2009)

The transition from the traditional software development models to the agile models is demanding. Challenges occur for instance in the organizational culture, competence, communication, tools, and management style. (Nerur, 2005) Thus, the social factors are inevitably involved in the transition. The major changes occur for instance in the communications and co-location. Multiple agile methods emphasize the co-location of the team. However, the team members may dislike the open office environment in terms of the privacy issues and the lack of personal space. (Law & Charron, 2005)

There are multiple approaches to the agile transition. The agile software development models include both top-down and bottom-up solutions. Thus, the transition should also involve both viewpoints. A frequently involved approach is to encourage communication, learning, and self-organization. In addition, transition teams or coaches are involved. Performing the transition as a revolutionary change is risky. The common approach is to initially launch the transition only with part of the organization. (Cohn, 2007)

2.1.7 Extreme Programming and Pair Programming

Extreme Programming (XP) defines some preferred practices for software development. However, the practices have been defined prior to Extreme Programming was introduced. The brilliance of XP is in the manner the practices are linked together. Whereas XP presents the engineering practices, the Scrum framework mostly supports on the management issues. Thus, Scrum and XP can be combined. However, some of the practices will overlap. (Abrahamsson et al., 2003; Kniber, 2007) The practices of XP include a planning game based on the customer demand (Beck, 1999) but also the incremental design work is emphasized (Kniber, 2007).

The practices of Extreme Programming include continuous integration, i.e., the new code is integrated into the system with frequent intervals. The code is collectively owned. Also, Pair Programming (PP) is involved. The concept in Pair Programming is that two engineers develop the same code on the same computer. (Beck, 1999; Kniber, 2007) Kniber (2007) states the Pair Programming to be enhancing the quality of code. However, according to a research by Hulkko & Abrahamsson (2005), the statement does not hold in all cases.

Some of the organizations utilizing Extreme Programming have developed a combination of rules or standards that have been identified as useful practices (Beck, 1999; Kniber, 2007). Test Driven Development (TDD) can be also applied in Extreme Programming. TDD implies that the testing begins prior to the coding is completed. The coding activities are driven by the findings from the tests. (Kniber, 2007)

Extreme Programming also emphasizes open work space and developing products in frequent releases (Beck, 1999). These practices are similar with the principles of the Scrum framework (Deemer et al., 2008; Schwaber, 2007; Sutherland, 2004).

2.2 The Scrum Framework

Next, the chapter describes the basics of the Scrum framework. First the overall procedure is discussed with the help of a figure of the framework, followed by an introduction of each artifact, role, and meeting. At the end of the chapter, the velocity of development and the definition of done are discussed.

2.2.1 Overview of Scrum

Scrum is a framework that can utilize various processes; it is not a process model itself. Scrum employs the iterative and incremental approach. The purpose is to control risk while being able to adapt to changes. There are three legs in the framework; transparency, inspection, and adaptation. Transparency enhances the visibility in all directions, i.e., to the scrum roles and to the corporate management. The inspection leg addresses identifying unwanted variance. The purpose of the adaptation leg is to enhance the means of working to increase productivity and quality. (Schwaber, 2009)

The traditional models of software development suggest that the requirements can be predefined. As opposite, the Scrum framework emphasizes the changing nature of the customer demands. Scrum is an ‘all-at-once’ model, i.e., all of the development activities are executed simultaneously. (Sutherland, 2001) Hence, also testing is integrated into the iteration, which leads to need for optimizing the testing with the aid of automation (Kniber, 2007; Santamaria, 2007). The term for iteration in the Scrum framework is ‘sprint’. The sprint is time-boxed, i.e., the duration is fixed from one to four weeks. The duration is suggested being the same for all sprints and it may not be extended within the sprint. (Deemer et al., 2008; Schwaber, 2009)

Scrum states that instead of releasing a wide base of new features in one release, the functions are released sequentially. Thus, the time-to-market for a single function is decreased by breaking a major release project to shorter cycles. Hence, the flexibility to adapt to the changing customer requirements is enhanced. The requirements are stated in form of user stories. (Santamaria, 2007)

Scrum includes four artifacts; the product backlog, the sprint backlog, the release burndown chart, and the sprint burndown chart. The roles involved are the scrum master, the product owner, and the scrum team. In addition to the development work within the sprint, the people interact in the sprint planning meeting, the release planning meeting, the daily scrum meeting, the sprint review, and the sprint retrospective. (Deemer et al., 2008; Kniber, 2007; Schwaber, 2009)

The Scrum framework, visualized in Figure 6, is initialized by inputs from the customers, the teams and other stakeholders. All of the features to be implemented are listed in the product backlog in form of user stories and requirements. In the sprint planning meeting, the high-priority items are selected from the backlog to be implemented in the next sprint. The number of selected items is a function of the estimated effort needed for implementing the items and the estimated velocity of the team. The selected items, i.e., the selected user stories, are split into tasks and placed in the sprint backlog. (Deemer et al., 2008; Kniber, 2007; Schwaber, 2009)

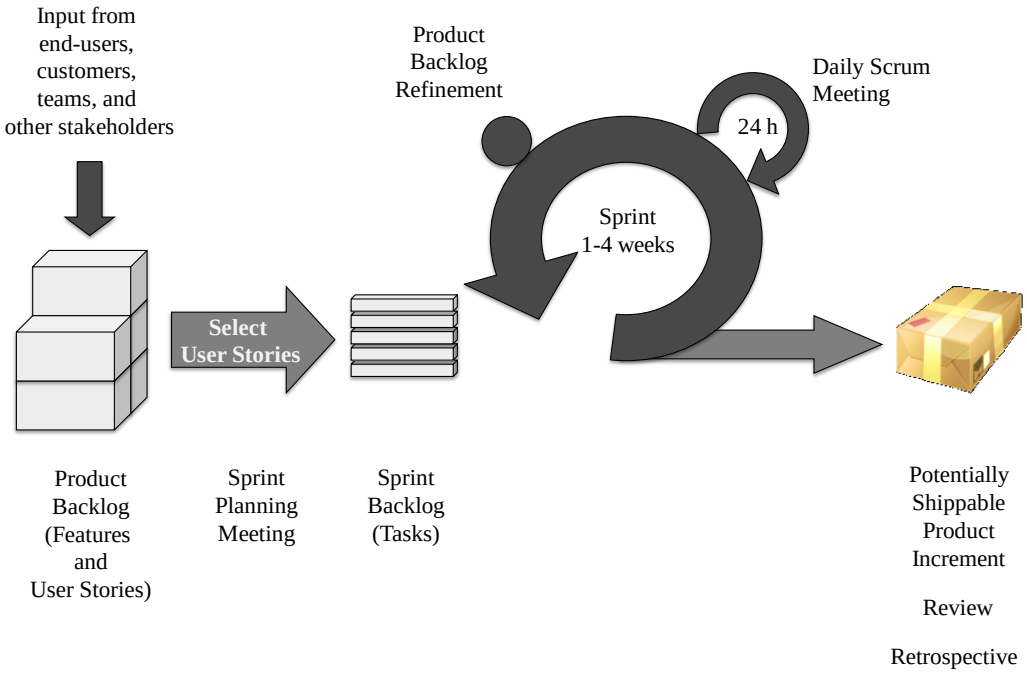

Figure 6. The Scrum Framework (modified from Deemer et al., 2008)

The sprint includes all of the development tasks, e.g., coding, testing, and writing the customer documentation. Over the sprint, a daily scrum meeting is held for discussing the current progress and impediments. The outcome of the sprint is a potentially shippable product increment. Thus, each sprint should provide a working functionality for the product. At the end of the sprint, the implemented feature is reviewed. In addition, a meeting called 'retrospective' is held to identify the key success factors and failures in the sprint. (Deemer et al., 2008; Kniber, 2007; Schwaber, 2009)

Figure 7 illustrates the sprint timeline. The sprint begins with a half-day sprint planning meeting. As already described, the team selects items for the sprint from the product backlog in the planning meeting. A daily meeting for enhancing the coordination and communication inside the team is held at the same time each day. The sprint ends to a sprint review and sprint retrospective. The review is for discussion and to demonstrate the increment, whereas the retrospective is for discussion regarding the process in the sprint. All of the meetings are time-boxed, i.e., the duration is fixed. (Deemer et. al, 2008; Kniber, 2007)

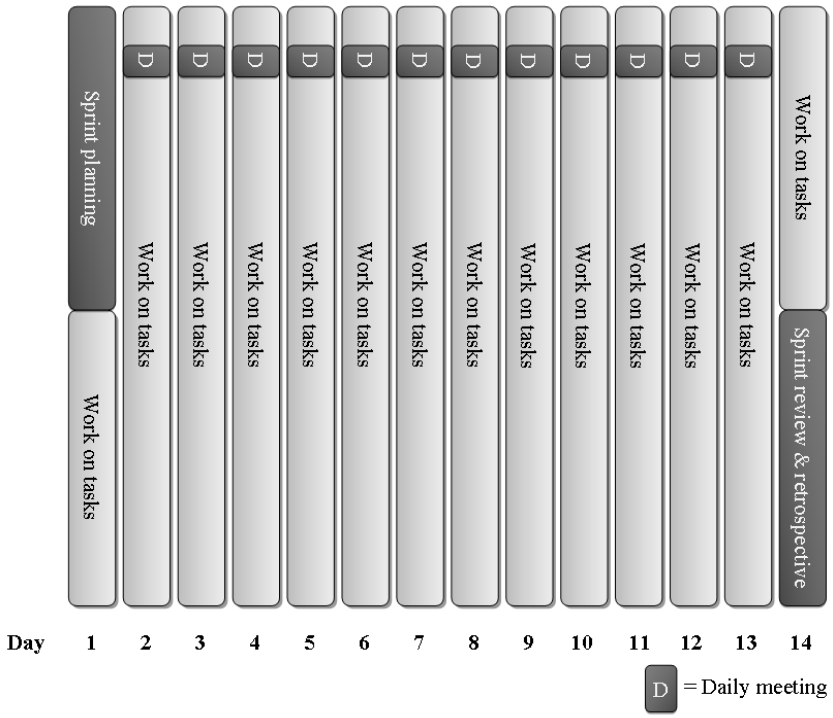

Figure 7. Timeline of the sprint (modified from Eskelinen et al., 2010)

The sprint can also be cancelled, e.g., if the sprint goal becomes obsolete. The product owner decides if the sprint is to be cancelled. However, this rarely happens since the sprint length is only from one to four weeks. It is unlikely to face changes in the demand over one sprint. (Schwaber, 2009)

One of the key features in the Scrum framework is the rapid feedback. Since coding and testing are executed simultaneous, the faults are detected earlier than in the Waterfall model. To reduce the length of the feedback loops, transparency is needed. The earlier the fault is detected, the cheaper and less complex it is to fix. The cost of fixing faults grows with higher than linear pace as a function of time. It is important to note that the new products as well as the new product increments face technical, architectural, and business risks. (Eskelinen et al., 2010)

All the development activities cannot be completed within the sprints. The features are tested by the team but additional tests are to be executed to secure the system level quality. The acceptance test is a phase between the sprint and the release. The team passes the completed features to the testers that are not in any scrum team. The scrum-external testers verify that the function is working as a part of the system combined from the outputs of all of the scrum teams. If a user interface is involved, manual testing is executed in addition to the automated testing. (Kniber, 2007)

The results of the Scrum framework have been positive. For example at Yahoo! the productivity increased by 68%, adaptability by 63%, cooperation by 81% and morale by 52% (Deemer et al., 2008). At Nokia Networks in 2006, almost 70% of the employees stated that they do not want to return to the previous means of working (Vodde, 2006).

The term for the faulty implementation of the Scrum framework is ‘ScrumButt’. It refers to a situation in which an organization believes that it is utilizing Scrum but actually the truth is opposite. To avoid ScrumButt, the Nokia test can be utilized. The test includes multiple-choice questions to an organization. Points are gained depending on the selected answers. (Sutherland, 2008) Appendix B presents the questions and answers along with the rules of gaining points in the test.

The faulty Scrum implementation can easily be implemented without making significant changes to the current means of working. The risk is that the framework is masked on top of the old means of working. Hence, the benefits of Scrum are not achieved. In this case, Scrum may be discarded by a management decision since the actual results were invisible. (Koskela, 2009)

2.2.2 Product Backlog and Prioritization

The product backlog (PB) is similar to a prioritized queue of tasks (Larman & Vodde, 2009). The product backlog includes a list of the features, enhancements, and needed bug fixes. These items include a description, a priority, and an estimation of the workload. The items are written in form of user stories, except in a case of technical improvements that might be impossible to write in such a manner. The items are split into tasks in the sprint planning meeting. (Schwaber, 2009)

A user story is a description of functionality from the user or customer perspective. The other terms related to the items are ‘epic’ and ‘theme’. An epic is a large user story whereas a theme is a collection of user stories related to each other. The proper user stories include multiple perspectives, recognize the variation of the customer demands, involve common attributes to all customers, clarify the common goal with the customer, and focus on the users instead of the system attributes (Cohn, 2008a, 2008b).

The product backlog is sorted in order of priority. The topmost items are detailed whereas the items at the bottom can be unspecified since they are not implemented in the next sprint. The backlog includes both minor and major sized items (Kniber, 2008; Schwaber, 2007, 2009).

Table 2 visualizes a product backlog. All of the items are linked to a wiki-page, which includes a detailed description of the item. Also the initial estimates for value and effort needed are stated in the backlog. The remaining workload is updated to the product backlog constantly. In the case of Table 2, for instance, the first two items are completed in the first sprint, third item in the second sprint and the fourth item in the third sprint. The estimation of effort needed for a single item might change between the sprints. (Deemer et al., 2008) Thus, the total amount of work to be done is dependent on the changes in the estimates for the workload and work performed in the previous sprints.

Table 2. An example of the product backlog (modified from Deemer et al., 2008)

Priority	Item	Details (Wiki URL)	Estimate of Value	Initial Estimate of Effort	New Estimates of Effort Remaining at end of Sprint				
					1	2	3	4	...
1	As a buyer, I want to place a book in a shopping cart (UI sketches on wiki)	www.firm.com/PB/1	7	5	0	0	0		
2	As a buyer, I want to remove a book from the shopping chart	www.firm.com/PB/2	6	2	0	0	0		
3	Improve transaction processing performance (target performance metrics on wiki)	www.firm.com/PB/3	6	13	13	0	0		
4	Investigate solutions for speeding up credit card validation (target performance metrics on wiki)	www.firm.com/PB/4	6	20	20	20	0		
5	Upgrade all servers to Apache 2.2.3	www.firm.com/PB/5	5	13	13	13	13		
6	Diagnose and fix the order processing script errors (bug ID 14823)	www.firm.com/PB/6	2	3	3	3	3		
7	As a shopper, I want to create and save a wish list	www.firm.com/PB/7	7	40	40	40	40		
8	As a shopper, I want to add or delete items on my wish list	www.firm.com/PB/8	4	20	20	20	20		
...	...	...	...	...	...	...	...		
			Total	**537**	**580**	**570**	**500**		

The product owner is responsible for the product backlog. The backlog is to be available and prioritized all the time. The backlog is never complete since new items are added on demand. Thus, the product backlog is dynamic and exists as long as the product exists. The product backlog reacts to the changes in the market conditions, technology, and business. Since there is a possibility of changes, it is convenient to describe only the highest priority items in detail. The product backlog re-factory process is called ‘grooming’. (Schwaber, 2007, 2009)

The proper prioritization is one of the key success factors for the Scrum framework implementation. The priority is a function of, e.g., business value, costs, risks, knowledge creation, and dependencies. (Eskelinen et al., 2010) Other people may add stories to the product backlog but the product owner is performing the prioritizing activities. As opposite, the team and only the team, estimates the workload of items since the teams often the best knowledge of the needed workload. (Kniber, 2007)

The Scrum framework does not state which prioritization technique to use. Cohn (2008a) identified four techniques that can be utilized. First, Kano analysis involving users will reveal which features are mandatory, which increase the satisfaction linearly, and which are the features that users do not know they want until they receive it. Second, themes can be screened with selection criteria by experts. Third, themes can be scored by adding weights to the screening selection criteria. Fourth, a relative weighting can be utilized for estimating the costs, value, and impact. These methods can be utilized as input to a comparison matrix of features. The matrix can focus for example on costs, net present value (NPV), internal rate of return (IRR), needed effort, or payback time. (Cohn, 2008a)

Also, the workload is to be estimated. An option is to utilize the planning poker. Each team member is provided with a deck of 13 cards including cards with numbers 0, ½, 1, 2, 3, 5, 8, 13, 21 (or 20) , 40, and 100. In other words, the Fibonacci numbers from one to thirteen are utilized with extra numbers for minor and major workloads. The numbers represent the relative workload of items, i.e., the number is not for example the amount of man-hours. There are no numbers between 20, 40, and 100 since too accurate numbers provide a false sense of accuracy. Additionally, there is a card with question mark and a card with a coffee cup. Intuitively, the card with a question mark indicates that the team member does not have an estimate for the workload whereas a coffee cup means that it is time for a brake. All of the team members show one card simultaneously with the other team members. (Eskelinen et al., 2010; Kniber, 2007) For the items selected for the sprint, the estimate for story points may not be changed (Microguru, 2008).

The items need to be proper sized. If the items are minor, there is a risk of involving micromanagement. However, large items lead to the only partially completed stories and estimating the number of stories to be selected for one sprint becomes difficult. For instance, it is difficult to decide the number of items to select if the team completes 70 story points per sprint and the highest priority items are estimated as 40 story points each. (Kniber, 2007)

Since the needs of the stakeholders of the product backlog vary, the backlog needs to include different views. For instance, there can be views of the product, release, project, and teams. One stakeholder might be a team that evaluates the new ideas. In one company, a concept of ‘opportunity team’ was established to evaluate the ideas and to insert the potential ones into the product backlog. (Nahor & Katzav, 2009)

2.2.3 Sprint Backlog and Task Board

In the sprint planning meeting, the team selects the topmost, i.e., the highest priority items, from the product backlog. They commit to implement the items in the next sprint. The number of selected items is a function of the number story points the team has estimated for items and the estimated velocity of the team. (Kniber, 2007; Schwaber 2009) Only the team is mandated to modify the item selection within the sprint (Schwaber, 2009).

The selected items are split into tasks and placed in the sprint backlog. The tasks should be detailed. The proper size of a task is four to sixteen man-hours. Thus, the tasks are measured in hours whereas the product backlog items are measured in relative story points. The tasks include information regarding the amount of work to be done. The table is updated on a daily basis. (Schwaber, 2007) Koskela (2009) suggests linking the sprint backlog with the product backlog to trace the progress of the product backlog items.

The sprint backlog can be located, e.g., a table in a software tool. The last row at the table states the remaining work in the current sprint. The figures are involved in drawing the sprint burndown chart. (Deemer et al., 2008; Kniber, 2008) Table 3 illustrates an example of the sprint backlog that is updated on a daily basis.

Table 3. An example of the sprint backlog (modified from Deemer et al., 2008)

Product Backlog Item	**Sprint Task**	**Volunteer**	**Initial Estimate of Effort**	**New Estimate of Effort Remaining at the end of Day**						
				1	**2**	**3**	**4**	**5**	**6**	**...**
Item 1: As a buyer, I want to place a book in a shopping cart	Modify database	Sanjay	5	4	3	0	0	0		
	Create webpage (UI)	Jing	3	3	3	2	0	0		
	Create webpage (JavaScript logic)	Tracy & Sam	2	2	2	2	1	0		
	Write automated acceptance test	Sarah	5	5	5	5	5	0		
	Update buyer help webpage	Sanjay	3	3	3	3	3	0		
	...	...	...	...	...	...	...	...		
Item 2: Improve transaction processing performance	Merge DCP code and complete layer-level tests	Jing	5	5	5	5	5	5		
	Complete machine order for pRank	Tracy	3	3	8	8	8	8		
	Change DCP and reader to use pRank http API	Julia	5	5	5	5	5	5		
...	...	...	...	...	...	...	...	...		
		Total	**50**	**49**	**48**	**44**	**43**	**34**		

The first column indicates the product backlog items the team has committed to. The item is split into tasks in the second column. The tasks are assigned to team members. In addition, an estimate for the effort needed is stated. In the example, the total estimate for needed effort is 50 hours. (Deemer et al., 2008)

Even if the team utilizes software to manage the sprint backlog, it is beneficial to utilize a physical scrum task board. The board is typically visible for all and enhances the interaction in the daily scrum meetings. The daily meeting is the proper time for updating the task board. Each white paper on the board represents a larger area of tasks. Each task is written to a sticker including the estimated effort needed, e.g., in hours. (Kniber, 2007) Figure 8 presents an example of a sprint task board.

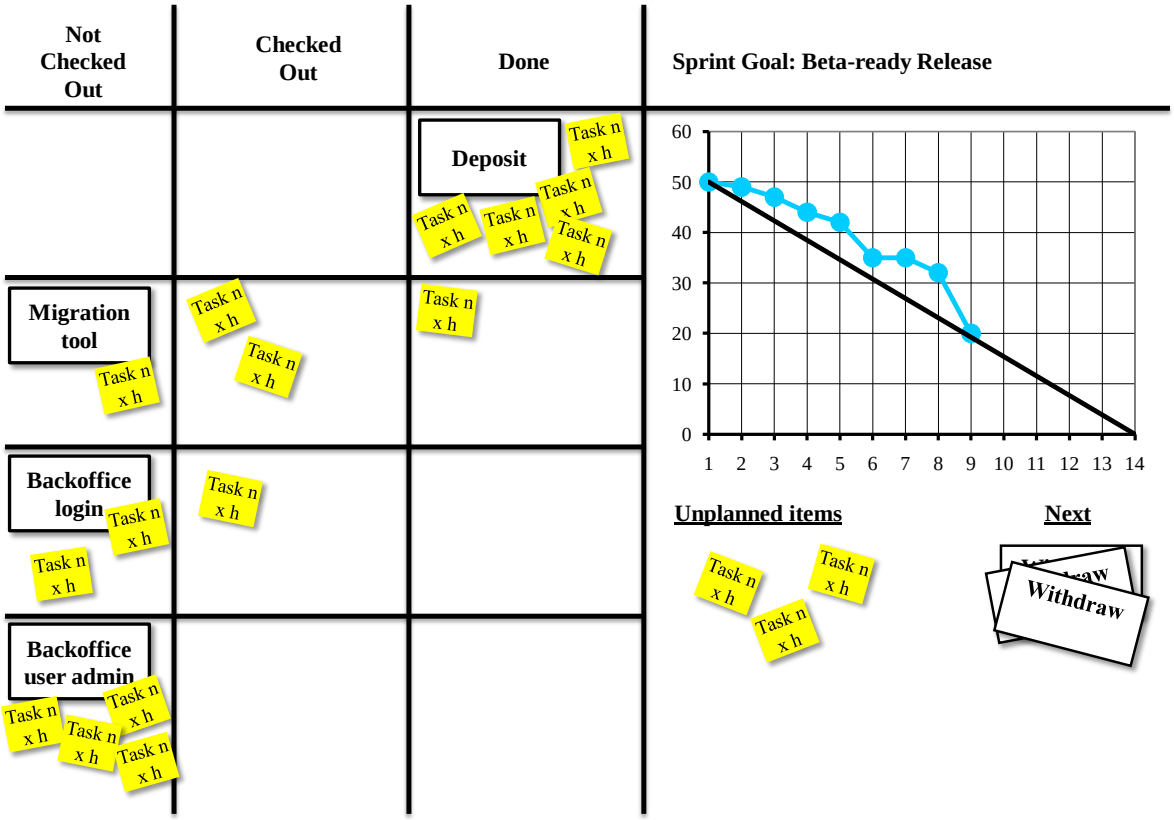

Figure 8. The sprint task board (modified from Kniber, 2007)

The first column on the board is the place for tasks that are not checked-out from the sprint backlog, i.e., tasks that no one is working on the current day. The second column is for the checked-out tasks, i.e., the tasks someone is working on the current day. The third column is for completed tasks. Thus, the stickers wander from left to right. Once all of the stickers related to one area are moved to stage 'done', the white paper is moved to the same column. The last column reveals the common sprint goal, and the sprint burndown chart, which is updated manually by hand. Also, the done items that were not planned for the current sprint are visible in the board. In addition, there is a place for items to be completed in case the team is ready with the selected items prior to the end of the sprint. (Kniber, 2007)

In the case of Figure 8, the team has completed the tasks of the deposit. In addition, they are progressing on the migration tool and back-office login tasks. This is the wanted behavior since the topmost items are with the highest priority. The team has also completed three unplanned items. (Kniber, 2007)

The task board can reveal warning signs. If the done and checked-out items are on the bottom, the team is focusing on the items with low priority. Also, if the majority of the progress is in the unplanned items, the team is not focused. (Kniber, 2007) In addition, if all of the items are checked-out but not completed, the team is not focused – they are performing the tasks only partially (Kniber, 2008).

2.2.4 Burndown Charts

There are two burndown charts in the Scrum framework; the sprint burndown chart and the release burndown chart. The graphs are called burndown since they reveal the workload left plotted as a function of time in a downward sloping curve. Both charts are constantly updated to form an estimate for the remaining workload. (Deemer et al., 2008; Kniber, 2007) The sprint burndown chart should be placed on the physical scrum task board to enhance accessibility and to utilize it in the daily meetings (Sutherland, 2009). Figure 9 visualizes an example of a sprint burndown chart.

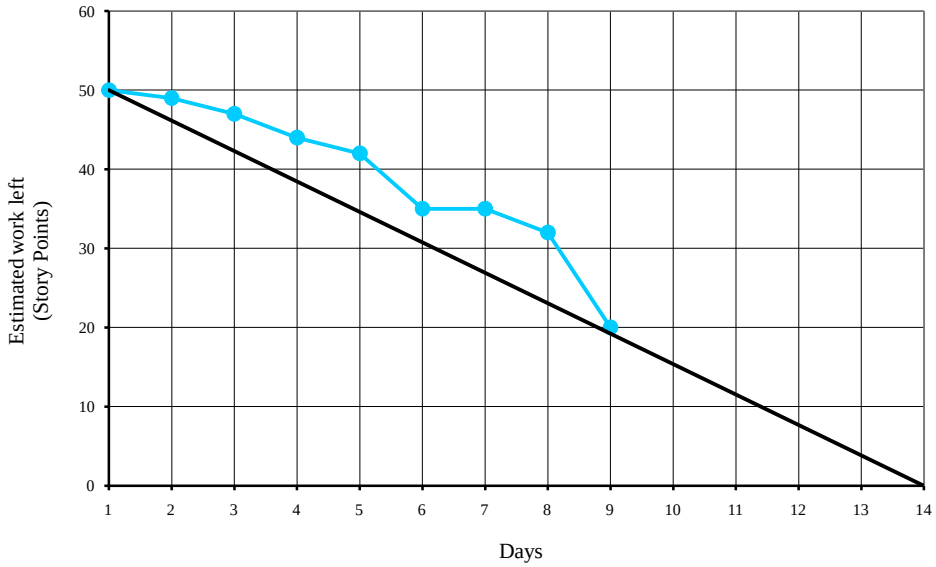

Figure 9. The sprint burndown chart (modified from Deemer et al., 2008)

The above chart presents a burndown chart of a sprint that lasts for 14 days. The chart is drawn in (estimated work left, time)-coordinates. The team may decide the unit for the vertical y-axis. The options are man-hours or story points. Each day, the current status is marked to the chart with a dot. A line is drawn from dot to dot. This line is the burndown line. Also a linear line, called 'the idealized line', is drawn. The line represents a linear progress from day one to the end of the sprint. Thus, the line begins from the point where the whole work is to be done at day one and ends to the point in which no work is left at the last day. In the example, 50 story points have been selected from the product backlog for the current sprint. According to the linear line, the point of no work left is reached at the last day of the sprint. (Deemer et al., 2008)

The focus is on the estimated remaining workload, not on the completed work. The chart is named as burndown chart since it reveals the amount of work still to be done. (Deemer et al., 2008; Kniber, 2007) In Figure 9, since the burndown line is above the idealized line, it can be interpreted that the team was first behind the estimation. However, the idealized line is reached at the day nine, which is also the current day of the sprint. It can also be interpreted that the work to be done is 20 story points. To enhance the accuracy of the chart, the team must define the criteria for stating the task or user story done, i.e., decide which actions are to be performed (Deemer et al., 2008; Kniber, 2007; Schwaber, 2008).

There are two different warning signs; there is too many or too few items selected from the product backlog to the sprint. If the burndown line stays significantly above the idealized line, i.e., the team is not performing as many story points as estimated, there are too many items selected for the sprint. As opposite, if the burndown line is significant below the idealized line, i.e., the team is performing more story points than estimated; too few items were selected from the product backlog. The remedy is, for instance, to deselect or select more items. (Kniber, 2007) Figure 10 visualizes the warning signs.

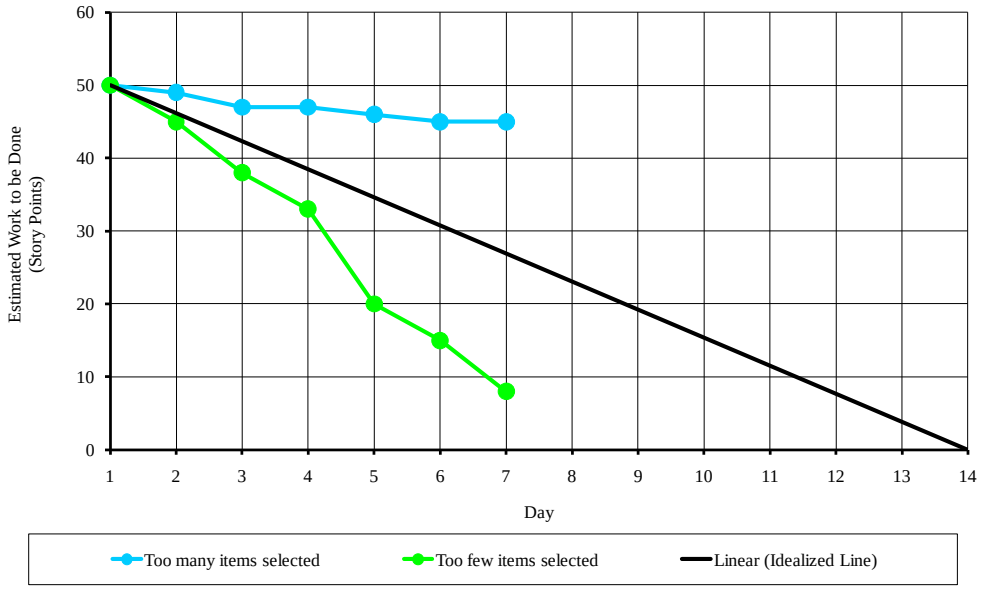

Figure 10. Sprint burndown chart warning signs (modified from Kniber, 2007)

The other chart in the Scrum framework is the release burndown chart that reveals the work still to be done in the release. There is similarity between the charts. However, the release burndown focuses on requirements, not on tasks. The unit for the time, i.e., x-axis, is sprints instead of days. (Deemer et al., 2008) It can be concluded that the sprint burndown is a status indicator for the sprint backlog tasks, whereas the release burndown is a status indicator for the product backlog items.

2.2.5 Scrum Roles

The Scrum framework includes three roles; the scrum team, the scrum master, and the product owner. There are no project managers. The team from five to nine professionals is the core of Scrum. Since the purpose is to complete a potentially shippable product increment in the sprint, the team is cross-functional, i.e., the members are familiar with tasks from design to coding and testing. The members have specialized skills but yet they all need to share the skill of understanding the process of creating product increments from the backlog items. (Deemer et al., 2008; Schwaber, 2009)

The team is an owner of both the sprint backlog and the process (Eskelinen et al., 2010). The whole team is to be aware of the basics of Scrum, thus, a course might be needed prior to implementing Scrum (Stuart, 2009). The teams should be formed based on the business needs (Nahor & Katzav, 2009) or features (Larman & Vodde, 2009, pp. 149-155), not based on the architecture of the product.

To enhance cooperation, the team should be seated in the same room. Everyone can see the task board and other team members. In addition, the members can discuss to each other without leaving their own desk. (Deemer et al., 2008, Kniber, 2007)

One of the fundamentals of Scrum is that the team is self-organizing. Thus, the team members decide themselves the means to create a product increment from a user story. The management and the scrum masters are not allowed to decide the procedure. (Schwaber, 2009) The team decides the stories they commit to and the means of successful implementation (Deemer et al., 2008).

The performance of a team is higher than the sum of the performances by the same number of individuals. However, if the team structure is changed, the performance may decrease. The key elements for the performance are the common sprint goal, collective responsibility, and cooperation. In addition, the team values and culture should be strong. Thus, the team needs to identify improvements to their practices. That is enhanced by the retrospective at the end of the sprint. (Eskelinen et al., 2010)

The scrum master (SM) is the mentor of the team and supports adopting Scrum. She or he is not managing the team since the team is self-organizing. The responsibilities also include enhancing the self-management and cross-functionality. (Schwaber, 2009) The scrum master supports the team in achieving business value (BV) and protects the team from outside interfaces (Deemer et al., 2008). In addition to guiding the team, the scrum master is the facilitator of the key meetings. Overall, the scrum master is responsible for ensuring that the team members are able to proceed in their tasks. Also, the creativity and empowerment are enhanced by the scrum master. (Eskelinen et al., 2010)

The scrum master may come from any background. If the scrum master has previously been a manager, it is suggested that she or he will not become the scrum master for the same developers she or he previously managed. (Deemer et al., 2008)

The product owner (PO) defines the features to implement and is responsible for return on investment (ROI). The tool for the work is the product backlog. Also, the product owner is responsible for the product backlog being updated, visible, and prioritized all the time. (Deemer et al., 2008) In addition to the product backlog, the product owner is responsible for the product burndown chart (Schwaber, 2008).

The product owner can be defined as a gateway between the development teams and the business. She or he creates user stories based on the needs of the business units and defines the acceptance criteria. The product owner is not mandated to modify the selection of the product backlog items by the teams. However, she or he is mandated to stop and restart the sprint. (Stuart, 2009) The product owner should sit close enough so

the team members can easily visit but far enough not to dictate the team in the everyday work (Deemer et al., 2008, Kniber, 2007).

2.2.6 Release and Sprint Planning

One of the false myths regarding the agile methods is that there is no planning or documentation. In fact, the agile planning can be divided into six levels; strategy, portfolio, product, release, iteration, and daily. The starting point is usually the product level. (Eskelinen et al., 2010) If the standpoint is taken to be the product level, first the release is planned. The outcome is a high-level plan and goal. The release planning also forms a basis for the prioritization of the product backlog. Second, the sprint is planned in the sprint planning meeting. (Schwaber, 2009)

Figure 11 visualizes the multi-level planning. The broadest strategic views are the product vision and the product roadmap. The roadmap defines multiple releases mapped to a timeline including interconnections between the releases. In the sprint planning meeting, the high-priority, i.e., the topmost, stories from the product backlog are selected and split into tasks. Outcomes of the sprints are completed functionalities. (Stuart, 2009)

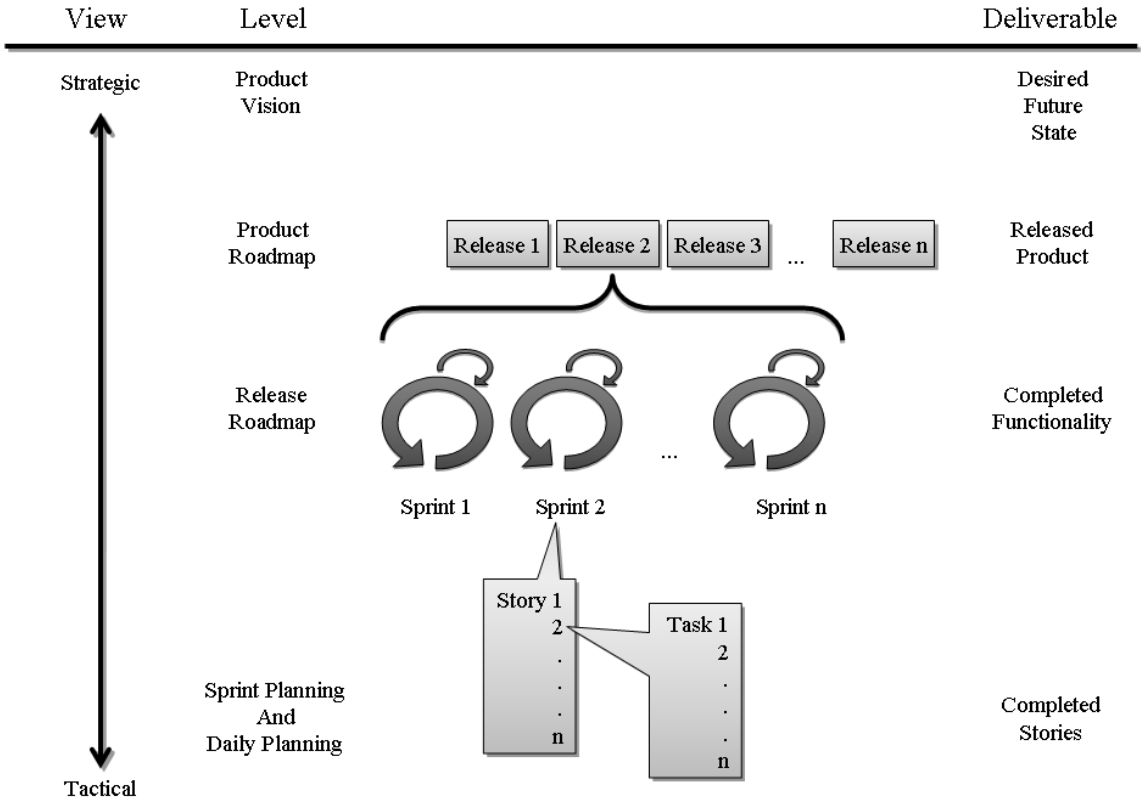

Figure 11. Multi-level planning (Stuart, 2009)

One purpose of the high-level planning is to identify dependencies and offer remedies to those. For instance, some features are to be completed prior to the product launch.

(Deemer et al., 2008) Koponen (2008) identified in his master's thesis that a product backlog tool needs to visualize dependencies between stories. In addition, he stated that the tool needs to support scheduling dependencies. The scheduling will be done as a roadmap, not with the help of Gantt charts. The stories might be depended in unidirectional or bidirectional manner. The weight of dependencies varies. (Koponen, 2008)

The lowest level of planning is the time-boxed sprint planning meeting. The maximum duration of the meeting is five percentage of the sprint length. Inputs to the sprint planning meeting are the team capacity and the product backlog (Schwaber, 2008). Thus, the product backlog is to be ready in forehand (Sutherland, 2009). The meeting can be divided into two parts. In the first part, the items from the product backlog are reviewed. The product owner is also present. In the second part, the team selects the items they commit to complete. In addition, the team forms the tasks, i.e., the selected product backlog stories are converted to the sprint backlog tasks. (Deemer et al., 2008)

To be aware of the number of the stories the team can select, the available working hours and the velocity is to be estimated. The available working hours is a function of the sprint length, the number of people involved, and the number of hours each team member can spend on the development process. (Deemer et al., 2008) The velocity calculations are discussed in Chapter 2.2.10. The outcome of the sprint planning meeting is a list of tasks in form of the sprint backlog budgeted with working hours (Schwaber, 2008). Also the sprint goal is defined in the sprint planning meeting (Kniber, 2007).

2.2.7 Daily Meeting

A daily scrum meeting lasts for approximately 15 minutes. The meeting is held at the same time and in the same place each day. The purpose is to improve communication and decision making in addition to problem solving. Only the team members and the scrum master are allowed to attend. Each member describes the work performed post to the last meeting, the work to be done prior to the next meeting, and the encountered blocking problems. The daily meeting is inspecting the progress in order to meet the sprint goals. (Schwaber, 2009) The meeting is important in terms of self-organizing the team (Deemer et al., 2008). If the daily meeting concept is not implemented successfully, the team is likely not aware of the current status and thus they encounter challenges achieving the goals they committed to (Schwaber, 2008).

In the meeting, the task board is updated. Done, work to be done, and tasks under development are written to stickers and placed in the proper columns on the board. Thus, the sprint backlog and the sprint burndown chart are updated. The whole team is involved in updating the current status and planning the next actions. (Kniber, 2007)

2.2.8 Sprint Review and Retrospective

At the last day of the sprint, two meetings are held. The first meeting is the sprint review, in which the team presents the results from the sprint and reflects on the results to the agreements on sprint planning. A demonstration is presented if possible. The second meeting is the sprint retrospective. In the retrospective, the team members discuss the process in the sprint. (Kniber, 2007; Schwaber, 2009) The purpose of the retrospective is to learn from each sprint and tailor the process accordingly (Stuart, 2009).

Inputs to the review meeting are, for instance, the backlogs, business conditions, and the completed product increments. Outputs can be the updated product backlog and next sprint goals. In the sprint review meeting, the unfinished stories are restored to the product backlog. In addition, if the team completed unselected product backlog items, these items will be removed from the product backlog. (Eskelinen et al., 2010) In addition to the team, the product owner and the scrum master are typically present at the meeting (Deemer et al., 2008; Kniber, 2007; Schwaber, 2009).

The retrospective is arranged to identify key success factors and potential areas of improvement in the process (Deemer et al., 2008; Kniber, 2007; Schwaber, 2009). The scrum master and team members are always present. The product owner presence is not mandatory. One option for organizing the meeting is to draw two columns on a whiteboard; another indicating the success factors and another indicating the potential areas of improvement. It is important that the team distinguishes which of these are related to Scrum and which would exist regardless of Scrum. (Deemer et al., 2008) The identified issues are prioritized and recognized in the future sprints (Schwaber, 2009). A suggestion is to focus only on one of the improvement areas per sprint (Kniber, 2007).

All of the teams need to arrange a retrospective meeting to enhance the operational excellence. If there are multiple teams, one person attends retrospectives arranged by different teams and thus enhances the learning from other teams by sharing the information. Another option is to write a report on the retrospective and share the paper with other teams. (Kniber, 2007)

2.2.9 Definition of Done

The underlying concept of Scrum is that each sprint provides an increment for the product. However, the term ‘done’ is to be defined by the team to share a common vision of the tasks. The done increment includes analysis, design, coding, testing, and, in some cases, documentation. These actions can be divided even further. For example testing can be divided into the unit, system, user, stability, performance, integration, and regression testing. (Schwaber, 2009) However, the definition may vary between tasks (Kniber, 2007).

Once the done is defined, intuitively also the complement, the undone, is defined. Since the amount of work to be done is nonlinear, the number of sprints becomes unpredictable. (Schwaber, 2009) If the team does not define done, multiple problems

can occur. For instance, there is no stable estimate for the velocity of the team. Unknown velocity leads to choosing too many or too few items for the next sprint. In addition, the product backlog burndown becomes inaccurate. Thus, the status is unknown and further planning is challenging. (Schwaber, 2008) The definition of done is to be controlled by the team (Kniber, 2008).

2.2.10 Velocity

Velocity is the term for the amount of story points completed over a sprint. In the sprint planning meeting, the team estimates how many story points they are able to complete in the next sprint. Thus, both the estimated and the actual velocity exist. The estimated velocity is involved in determining the number of items to be selected. The velocity data is important for the team. If the velocity is unknown, it is impossible to form a product roadmap with release dates. Hence, velocity is an important tool for the scrum team and for the product owner in terms of the different levels of planning. Figure 12 illustrates the difference between the estimated velocity (V_e) and the actual velocity (V_a). Note that the partially-done items are discarded in the velocity calculations. This is due the principle of developing a potentially shippable product increment within a sprint. (Kniber, 2007)

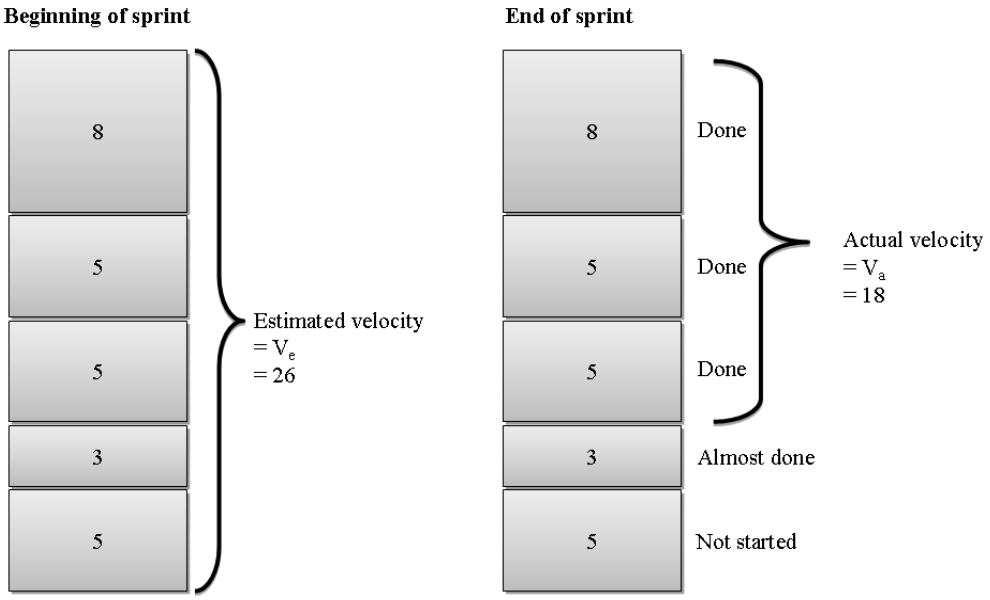

Figure 12. Estimated and actual velocity (modified from Kniber, 2007)

There are different techniques for estimating the velocity at the beginning of a sprint. First option is to estimate based on thoughts and feelings. Of course, this is inaccurate. The second option is to assume that the velocity equals the velocity in the previous sprints. However, the velocity is not constant. (Kniber, 2007)

The third option is to utilize information regarding the available man-days and a focus factor in estimating the velocity. First, the available man-days are calculated. In other words, the days spent, for instance, on other projects or in vacation are reduced from the

total number of days in the sprint. All hours spent at the office are not effective hours. Thus, there is a need for a focus factor. The factor is determined by dividing the actual velocity of the previous sprint by the available man-days in the previous sprint. Hence, the estimated velocity can be determined with equation

$$V_{e,n} = \sum_{i=1}^{k} (t_{i,\,at\,work} - t_{i,\,other\,projects})_n \times \frac{V_{a,\,n-1}}{\sum_{i=1}^{k} (t_{i,\,at\,work} - t_{i,\,other\,projects})_{n-1}}, \qquad (1)$$

where k refers to the number of scrum teams, $t_{at\ work}$ to the time an employee is at work, $t_{other\ projects}$ intuitively to the time spent on other activities than the development tasks in the scrum team, and $V_{a,\,n-1}$ to the actual velocity of the previous sprint. The factor on the left denotes the available man-days whereas the factor on the right denotes the focus factor. (Kniber, 2007)

It is useful to note that the time factor may utilize other units than man-days, for example man-hours, since the units cancel each other (Kniber, 2007). According to Sutherland et al. (2009), the velocity is typically estimated in terms of story points, never in terms of man-hours. Accuracy can be enhanced by increasing the time scope of the variables, i.e., by combining the figures from multiple past sprints to discard the deviation. (Kniber, 2007) Koponen (2008) suggests involving only a portion of the past velocities in extrapolating the estimated velocity, since the estimate can be more feasible if the sprints with the highest deviation are discarded. He suggests visualizing velocities with a bar chart.

According to Koponen (2008), the velocity calculations based on history data should also include an estimate for variation, i.e., instead of the date of release completion, a range of possible dates should be identified. Figure 13 illustrates the concept.

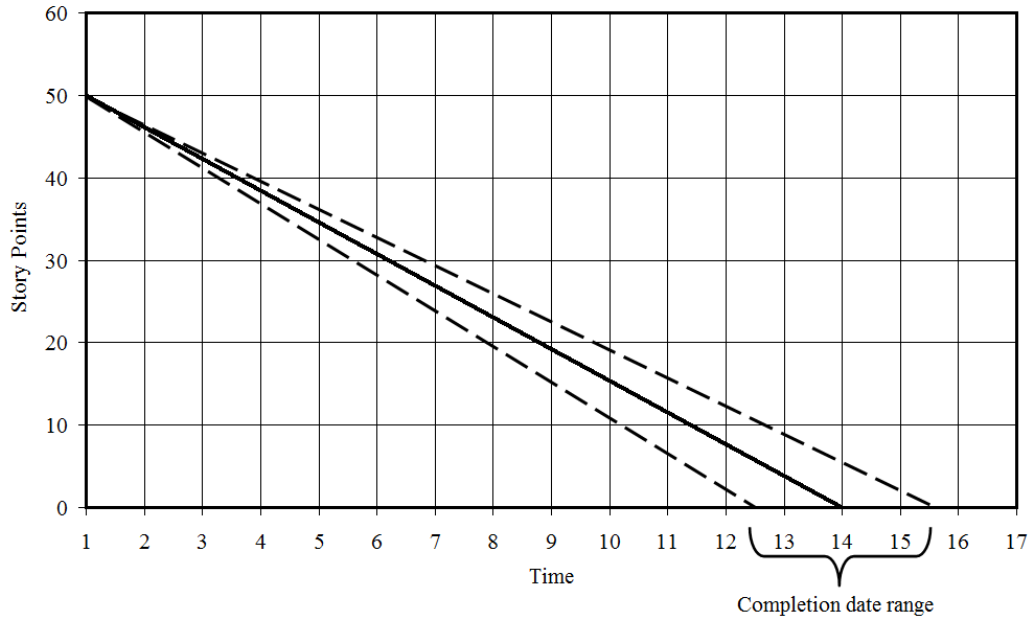

Figure 13. The range of completion dates (Koponen, 2008)

If the team overestimates the velocity, technical debt is generated. In other words, in case of overestimated velocity, the team will select too many items from the product backlog leading to failure in delivering. (Eskelinen et al., 2010)

The productivity is said to be higher with the Scrum framework than with the Waterfall model. Sutherland et al. (2009) define that hyper-productivity is reached once a scrum team performs with 400% higher velocity than the average waterfall team. The best scrum teams have achieved the velocity of 750% higher than in the Waterfall. (Sutherland et al., 2009) Often velocity increases with time since the focus factor increases once the team becomes familiar with Scrum (Sutherland & Altman, 2010). The teams improve most by solving their own problems and by self-organizing (Schwaber, 2008).

Sutherland et al. (2009) identified twelve key success factors for the hyper-productivity. These include, among others, enhancing the definition of done, mapping the dependencies between the tasks on the scrum board, estimating in story points instead of work hours, improving the readiness of the product backlog items, and changing the organization on a regular basis. In addition, the corporation needs to commit, trust, and be open. (Sutherland et al., 2009)

2.3 Innovations, State-Gates, and Metrics in Agile

Schilling (2004, pp. 216-224) argues that for an innovation to succeed, the fit with customer requirements is to be maximized, the development cycle is to be minimized, the development is to be performed parallel, and both the customers and the suppliers are to be involved. Thus, the innovations are to be managed. Dogson et al. (2008, pp. 1-10) introduce a term ‘Management of Technological Innovation’ (MTI) for that purpose. This chapter examines the state-gate process model for new product development (NPD). In addition, the combination of the agile software development models and the state-gate process model is discussed. Also, the links between agile, innovativeness, and metrics are discussed.

The state-gate process model by Cooper (2000, 2001) introduces gates that are utilized as decision points, as quality-control checkpoints, and as a path forward. The model emphasizes cross-functionality, built-in customer interaction, parallel activities, and defining the product early. (Cooper, 2000, 2001) Except the early definition phase, the fundamentals of the state-gate model are similar to Scrum (Deemer et. al, 2008; Schwaber, 2009). Cooper (2000) defines the model as follows: "A state-gate process is a conceptual and operational roadmap for moving a new product project from idea to launch". Figures 14 and 15 visualize the state-gate process.

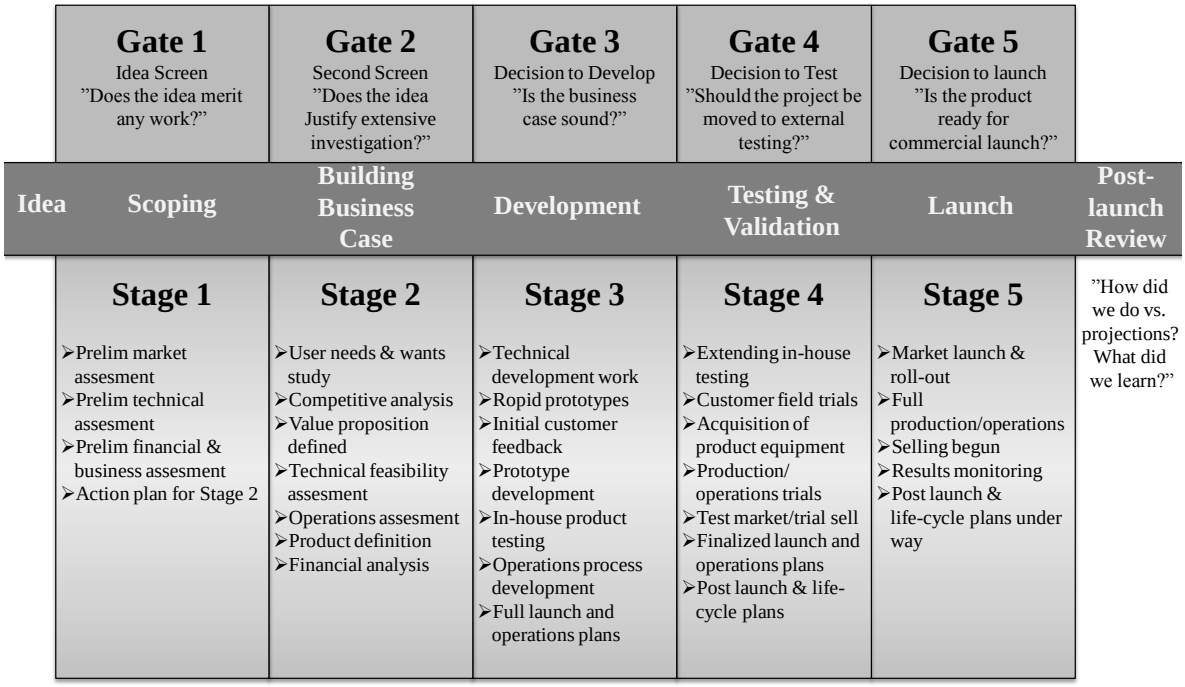

Figure 14. The state-gate process model (Cooper, 2000)

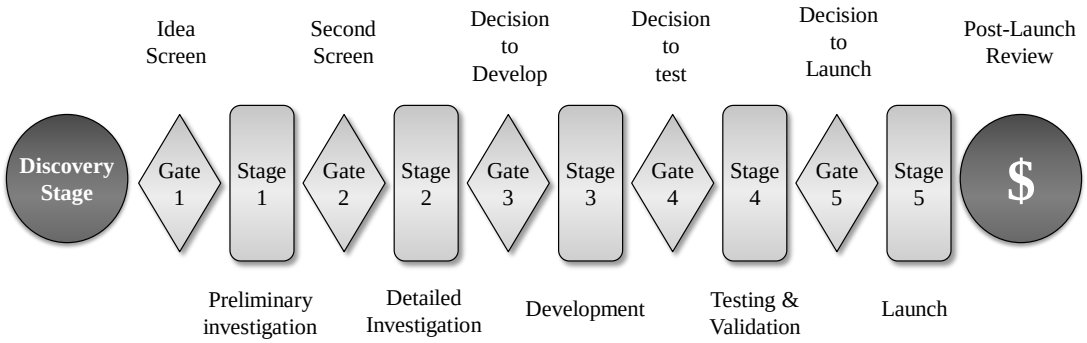

Figure 15. Flow of the state-gate process model (Cooper, 2001)

The state-gate model manages the new product development with a defined framework. First stage involves quick investigation, second stage building a business case and detailed planning, third stage development decision, and fourth stage testing. Finally, fifth stage involves launching the product to markets. (Cooper, 2000) However, the model does not fit as such – it is to be tailored for the needs of a company (Dogson et al., 2008, pp. 220-223; Schilling, 2004, pp. 224-226).

Karlström and Runeson (2005) discuss experiences of combining the agile methods with the stage-gate process model. The paper relies on case studies at ABB Automation, Ericsson Microwave Systems, and Vodafone. The study argues that the utilized agile methods reduce the length of the feedback loops and prevent the project deadline from affecting the scope of the agile teams. In addition, by combining these two methodologies, the teams were able to enhance the understanding regarding the system.

The major obstacle was identified to be the communication. With the help of the agile methods, the teams were able to focus on the day-to-day work control and micro-planning whereas the state-gate process helped in the long-term strategy and communication to other than developing parties, e.g., the marketing and the senior management. However, since the long-term strategy was formed by the management alone, the managers needed to contribute to the technical decisions. (Karlström & Runeson, 2005)

According to Larman and Vodde (2009), the state-gate process model is to be discarded if the Scrum framework is utilized. The reasons are that the state-gate model promotes large deliveries and thus delays are created, it focuses on individual products rather than the portfolio, and it transforms Scrum towards the Waterfall model. However, the brilliance of the state-gate model in keeping the process on track is recognized but it is stated that the gate-type of behavior at the end of each sprint, i.e., the sprint review, is equally effective. (Larman & Vodde, 2009, pp. 208-210)

In addition to avoiding the state-gate model, Larman and Vodde (2009) emphasize the need to avoid the parallel releases. Parallel releases, clarified in Figure 16, lead to a waterfall type of product development, i.e., early analysis. In this case, re-analysis is needed since the original analysis will not be feasible in the later phases. (Larman & Vodde, 2009, pp. 208-210)

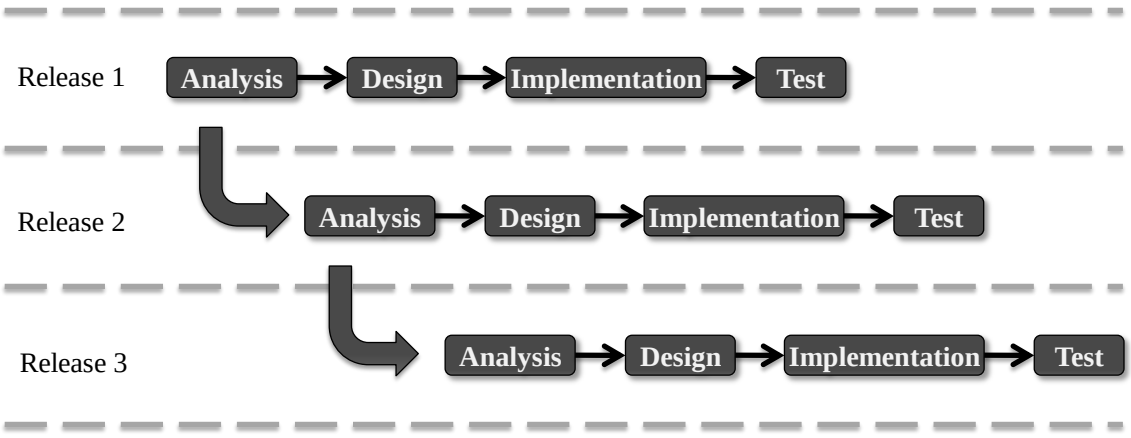

Figure 16. The parallel release development (Larman & Vodde, 2009, pp. 209)

From the innovation perspective, in Scrum, new ideas can be placed in the product backlog without immediate commitment to implement the item (Racheva & Daneva, 2008). The ideas are filtered in the frequent review sessions that the framework provides. Thus, the Scrum framework includes a built-in innovation management tool. The items in the product backlog can be prioritized based on business value and risk. The framework enhances the communication and transparency between different units of the organization. In addition, the feedback loops of the new product development are shortened. (Barton, 2009)

To manage the innovation and the product development, valid metrics are to be introduced. The data collection process is to be lightweight and the metrics need to add value. (Sulaiman et al., 2006) Racheva and Daneva (2008) divide the metrics into three categories; measurements at the code level, productivity metrics, and economic metrics. The code level metrics trace for instance the quality of code and the number of tested features. (Racheva and Daneva, 2008)

The productivity metrics already discussed in the thesis are burndown charts, velocity, and focus factor, i.e., the difference between done and planned work (Chapters 2.2.4 and 2.2.10). Also, time-to-market has been mentioned (Chapter 2.2.1). According to Koponen (2008), the amount of waste could be a metric. However, from the release management perspective, the estimate for the release date is especially interesting (Sulaiman et al., 2006).

Economic or financial metrics include for instance earned business value (EBV) Racheva and Daneva, 2008), break-even calculations (Racheva and Daneva, 2008), customer satisfaction (Tengshe & Noble, 2007), value delivered per release (Tengshe & Noble, 2007), return on investment (ROI) (Deemer et al., 2008), actual costs versus planned costs (Koponen, 2008), and cost per story point (Koponen, 2008).

Sulaiman et al. (2006) introduce a concept of ‘Agile Earned Value Management’ (AgileEVM), which was experimented in two projects. The combination of measures includes the velocity, the mean velocity, an estimate for the release date, an estimate for the number of sprints left, the burndown chart, the cost performance index (CPI) and the schedule performance index (SPI). The executive management and the team members were interested in CPI and SPI. The study suggests that the product owner or the scrum master do not benefit from AgileEVM compared with the burndown chart if they are not financially responsible. (Sulaiman et al., 2006)

As visible in the study regarding AgileEVM, different stakeholders demand different data (Sulaiman et al., 2006). Tengshe and Noble (2007) suggest formulating an approval matrix to identify the authorized persons at each decision point and to breaking down the metrics to different levels, e.g., to release or to feature level to enhance the benefits for different stakeholders.

With the help of metrics, real options analysis can be utilized. The concept of real options is to proceed with minor steps and to form only the needed decisions at each point. The method recognizes the uncertainty and the dynamic nature of the context by allowing incremental proceeding and by understanding the difference in the values of the options. At each decision point, work can be postponed or abandoned. Hence, features can be removed or added. The model includes similarities with the agile software development principles. (Racheva & Daneva, 2008)

2.4 Large-scale Implementation of Scrum

In this thesis, scaling the Scrum framework refers to the transition from the pilot scrum teams to utilizing the framework in a large extent in new product development. Next, options for scaling are discussed followed by identifying key success factors and challenges in scaling.

2.4.1 Options for Scaling

According to Sutherland (2009), the Scrum framework scales linearly. He suggests that the velocity of a scrum team stays linear even if the number of teams increases. As opposite, the performance of a waterfall team decreases while the number of teams increases. In most companies utilizing Scrum, the transition process towards agile started with a pilot project. (Sutherland, 2009) A pilot prior to scaling is less risky than starting the agile transition by involving the whole organization from the beginning (Cohn, 2007).

A six-step-approach can be involved to adopt the Scrum framework. First, the plans are defined, metrics are established, and the people to the pilot scrum teams are trained. Second, the framework is piloted for three to six months. Third, the expansion towards organization level begins by further training. Fourth, approximately a quarter of the organization is transformed to Scrum. Fifth, a combination of broad metrics is introduced regarding both the process and the new product development. In addition, the process is adjusted. Finally, Scrum is expanded throughout the organization. (Rally Software, 2009)

Implementing the Scrum framework will affect the organizational structure. The organization is transformed to as flat as possible, i.e., there will be fewer managers than earlier. (Koskela, 2009; Larman & Vodde, 2009, pp. 241-245; Sutherland, 2009) However, all of the previous structures cannot be destroyed. For instance, human resources and other supporting roles such as the IT support and the test tool maintenance will remain in separated units. The separated units related directly to development should be avoided. Indeed, an option is to utilize the scrum teams as the organizational units. If there are multiple teams developing the same features, it is beneficial to group the teams, e.g., based on the requirement areas, to enhance synergies, learning, and systems thinking. (Larman & Vodde, 2009, pp. 241-245)

According to Kniber (2007), virtual teams emerge easily. For instance, inside a large scrum team, some of the members might form virtual teams. As opposite, a virtual team might emerge based on two small-sized teams. A remedy is to perform an experimental team formation and adjust the formal team selection to be similar with the emerging virtual teams. (Kniber, 2007) To introduce the principles of the Scrum framework to each team, the pilot teams can be split in such a manner that all of the new teams include at least one member with Scrum experience (Cohn, 2007).

In the new product development project, faults are identified also in the released products. Fixing these disturbs the scrum teams. Thus, it is advantageous to establish a separated team to fix the faults on the legacy code rapidly and to let the scrum teams to focus on the new product development. However, expertise from the scrum teams might be needed to for the fixes. (Kniber, 2007)

The daily meetings of the scrum teams should be scheduled to be interleaving within the morning. With this kind of approach, the product owner can participate in all of the daily meetings since they are not organized at the same time and yet they are not distributed throughout the whole day. In addition, the team members can visit the daily meetings of the other teams if needed. The sprints should be synchronized to enhance the collaboration and to decrease the administrative overhead. (Kniber, 2007)

The ‘Scrum-of-Scrums’ is a concept for linking the teams. It is a weekly meeting in which all of the scrum masters discuss the items completed, impediments, and the cross-team concerns at the product level. (Kniber, 2007)

When it comes to the product backlog, the scaled Scrum framework will include for instance functional, non-functional, and infrastructure requirements that can be placed in the product backlog (Rally Software, 2005). A large product development unit may utilize one or multiple product backlogs. If only one product backlog is utilized, the needed skills should be stated in the items. With the help of this knowledge, the items from the product backlog can be selected by the teams with the proper skills. Once the synchronized sprints have come to end, the sprint review can be held jointly. Also, a joint sprint retrospective can be held. As a complement, a sprint retrospective is to be held for each team separately. (Larman & Vodde, 2009, pp. 176-177, 292)

If multiple product backlogs are utilized, an option is to introduce a concept of the area product backlogs. The interconnected teams are grouped to form areas. The product owner responsibility is distributed to multiple people, i.e., to the area product owners (APO). Each requirement area includes a backlog, i.e., the area product backlog. The areas are related to features and business, not to the architecture of the product. (Larman & Vodde, 2009, pp. 217, 241-249, 296-300) Figure 17 illustrates the scaled Scrum framework utilizing the product area concept.

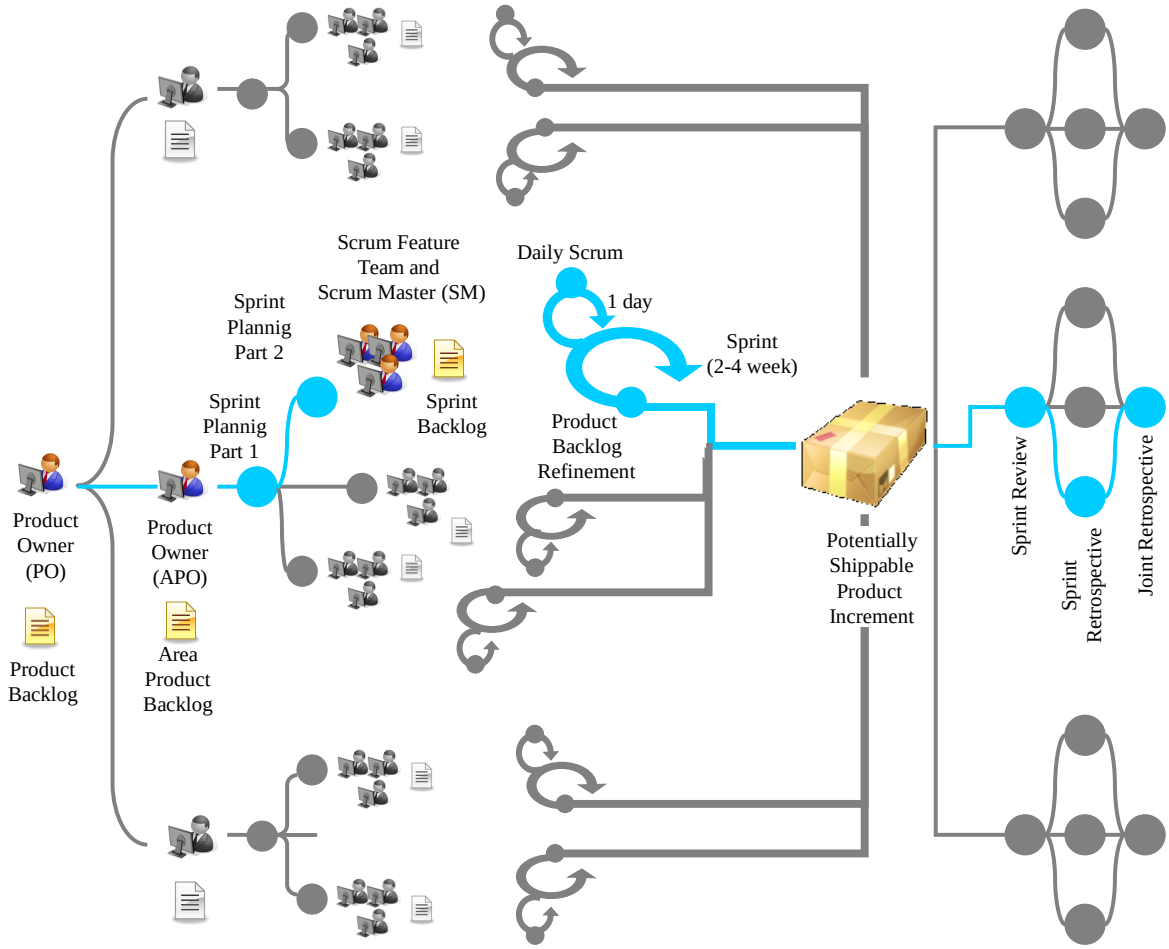

Figure 17. Scaled Scrum (modified from Larman & Vodde, 2009, pp. 299)

2.4.2 Challenges in Scaling

In a survey of The French Scrum User Group (2009), four major challenges for implementing Scrum were identified to be the interaction with the non-agile entities, the management involvement, estimation and planning, and identifying a product owner. In addition, challenges may occur in aligning the high level business objectives and the product portfolio while the silos of knowledge still remain intact (Kalliney, 2009).

As the complexity grows, the specialization and knowledge fragmentation occurs easily. Dependencies emerge, which can lead to conflicts between the teams. A team may need another team to perform a task to proceed in their own work even if the second team has different priorities. Hence, dependency management is to be established. (Babinet & Ramanathan, 2008)

The dependencies increase the complexity of a large-scale product backlog since the items are interconnected (Rawsthorne, 2008). If multiple backlogs are implemented, interconnections and thus dependencies emerge between the backlogs (Larman &

Vodde, 2009, pp. 296-300; Rawsthorne, 2008). However, if only one product backlog is implemented, it can include thousands of items (Rally Software, 2005).

Also, the product owner may become too busy leading to a bottleneck in the decision process. Thus, the prioritization of the product backlog is threatened and problems may occur for the product owner in understanding the items added to a wide range of features. As an outcome, delays emerge and direction of the project is blurred. (Lowery & Evans, 2007)

Last, to provide yet another perspective for the discussion, in the global corporations, the scaling of the agile methods involves distributed new product development. The challenges emerge, for instance, because of the time-zone difference, lack of co-location, cultural differences, and lack of transparency. (Paasivaara et al., 2008) In addition, the responsibilities regarding updating the product backlog may become unclear (Paasivaara et al., 2009).

2.4.3 Key Success Factors for Scaling

While scaling the agile methods from the pilot teams to the full extent, it is important to eliminate the silos of knowledge, for instance, with the help of Pair Programming (Kalliney, 2009) and to develop a common language (Lyon & Evans, 2008). In addition to enhancing the professional skills, the agile skills are to be enhanced. At Yahoo! it was estimated that an agile coach can save USD 1.5 million in the corporate expenditures per year by coaching ten teams each year. (Benefield, 2008)

Even tough there are multiple agile software development methods available each of them is to be tailored to the needs of the company. The management is to be aligned and the organization is to be adapted. The transparent feedback and internal lobbying of the agile methods are valuable in creating a positive atmosphere for the transition. In addition, the funding and the headcount are to be aligned with the transition. (Benefield, 2008)

In Chapter 2.4.1, it was identified that the product owner responsibility can be split. For instance, at British Broadcast Corporation (BBC), one product owner had too much work, which led to delays in the decision making and in keeping the product backlog updated. Thus, the tasks were divided for multiple people. However, the clear definition of the product ownership was emphasized. (Lowery & Evans, 2007)

The product backlog may include multiple hierarchy levels. For example, Lyon and Evans (2008) suggest involving one master product backlog and a minor backlog for each product. The product-level backlogs are to be linked to the master backlog to enhance the transparency of different levels. (Lyon & Evans, 2008) Multiple levels of the product backlog enable providing modifications on the views of the backlog based on the needs of different stakeholders. For instance, the items can be grouped to provide a high-level view for managers. However, the complex backlogs may lead to a situation, in which the completed story adds value to multiple places. (Rawsthorne, 2008)

If multiple levels of the product backlog are utilized, new attributes are introduced. For instance, the system and the backlog to which the item belongs are to be stated. Also dependencies are to be tagged. (Rawsthorne, 2008) The dependencies can be discussed in the meetings of the scrum masters, i.e., the Scrum-of-Scrums, followed by mapping of the dependencies (Lowery & Evans, 2007; Babinet & Ramanathan, 2008).

Babinet and Ramanathan (2008) suggest a framework for visualizing the inter-team dependencies. A circle is drawn to present each of the teams. Two arrows are drawn, the first from the team performing the task to the team that needs the task done, and the second towards the impacted team. A description of the task is added to each arrow. Once the team has committed to deliver the task, a box is drawn around the description. Since each of the dependencies is visualized, the critical spots and bottlenecks are intuitively visible. (Babinet & Ramanathan, 2009)

In addition, scaling affects the release management. Sutherland (2005) identified three types of Scrum; A, B, and C. The type A Scrum refers to sequential time-boxed sprints that are frequently utilized in the pilot phase. The type A Scrum is easy to implement and keeps the focus on the current tasks. However, it increases the time-to-market and generates delays for instance due to the downtimes between the sprints. Thus, the efficiency is to be enhanced. In the type B Scrum, the definition of new items begins in the previous sprint than the implementation. Thus, the downtime and the false sprint starts are eliminated. The type B Scrum provides an option for the hyper-productivity. A requirement for the type B Scrum is identified to be the full availability of the product backlog, i.e., the backlog is to be prioritized and ready for the splitting of the items at any time. (Sutherland, 2005)

Whereas the type B Scrum provides smooth transition from one sprint to another, the type C Scrum introduces a concept of overlapping sprints in which the teams work with multiple releases at the same time. The result is all-in-one streamline development. Automation and Scrum-of-Scrums are needed to support the development. The type C Scrum is suggested to deliver monthly releases for the selected customers and a quarterly release with major functionality to the whole market. (Sutherland, 2005)

Another approach to the large-scale release management is the release train by Leffingwell (2007). The train emphasizes approximately 60-days release cycles with two different releases; the internal release and the external release. The sprints are synchronized or otherwise the slowest component development delays the whole train. It is to be noted that the lagging does not automatically indicate low performance. It can also be due to technological risk or over-optimistic assumptions. Additionally, the synchronization eases the continuous system integration. Between iterations, the components are integration to the system. Post the internal or external release, the next sprint is for hardening, i.e., developing the product to even better. (Leffingwell, 2007) Figure 18 presents the release train concept.

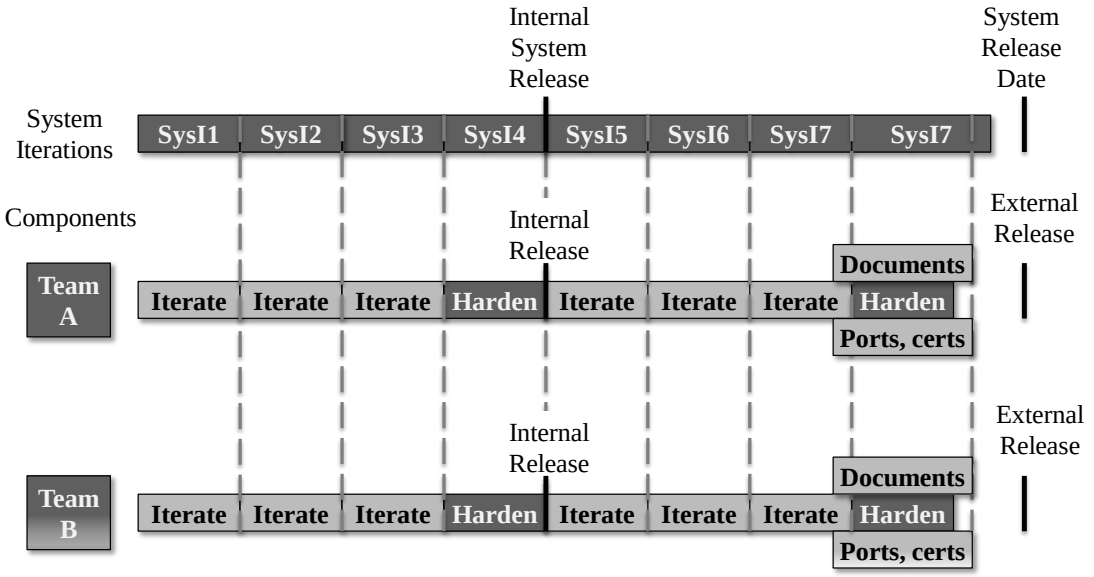

Figure 18. The agile release train (modified from Leffingwell, 2007)

Vähäniitty (2005) presents a framework, 'Cycles of Control', for connecting the strategic management and the software development. For each cycle, among others, the purpose, the time horizon, the interfaces, and the key decisions are characterized. There are seven cycles in the suggested framework. The cycles begin from a low level, i.e., the daily heartbeat cycle, the iteration, and the release project. The topmost four levels include the development portfolio management of current and planned development, the strategy of the individual product, the business unit level strategy, and finally the corporate strategy. The fourth cycle, the development portfolio management, connects the business and the development decision making process. (Vähäniitty, 2005)

2.5 Agile in the Telecommunications Domain

Since the thesis is written in the telecommunications domain, literature regarding other companies in the domain was examined. Additionally, other Ericsson sites were benchmarked from literature. The benchmarking of other companies and sites was divided into three parts; characteristics of agile in the domain, agile in the network vendors, and agile in the operators. Hence, from the market position perspective, both competitors and customers were benchmarked.

2.5.1 Overview of Agile in the Telecommunications Domain

The telecommunications domain is facing major changes, e.g., the raise of the mobile broadband and the mobile internet, leading to new types of services. Meanwhile, the interoperability and the standards are emphasized. The boundaries for the new product development are also defined by the regulatory rules and the technical limits such as the available range of spectrums for the radio connections. (Ericsson, 2009) There is no single dominant solution for the network technologies, instead, multiple technologies coexist (Kettunen, 2009).

The new product development in the telecommunications domain involves both software and hardware. Often the customer does not perceive the quality of the software as such – the expired quality is a combination of the qualities of both software and hardware. The embedded nature of the software generates dependencies and increases complexity. For instance, the lead time for developing the software and hardware are different. Hence, the hardware limits the date of completing the testing environment. Thus, in addition to the other external and internal dependencies of the software development, dependencies occur with the hardware. (Kettunen, 2009)

In his doctoral dissertation, Kettunen (2009) identified concerns regarding the large-scale embedded software development in the telecommunications domain. These include the uncertainty of the market, the development of different functions than the market demands, the unrealistic budgets, the inter-component or inter-group dependencies, the performance issues, and the lack of sufficient supporting computer infrastructure. In addition, the problems regarding the sufficient knowledge were identified. For instance, the software developers may lack the knowledge of the hardware, the architecture is complex, and the requirements change or are unclear. (Kettunen, 2009)

The new product development of the telecommunications systems is based on platforms and legacy systems. In addition, there are existing customer bases and installations in the markets. Thus, the complexity is high and dependencies occur for instance in the form of configuration compatibility. (Kettunen, 2009)

The telecommunications product development frequently involves a large organization. Thus, the implementation of the agile methods is to be tailored. For instance, experiences from Nokia and Motorola reveal that Extreme Programming is to be tailored to meet the needs of the company. However, also the organizational culture is facing pressure to adapt to the new means of working. In addition, the product architecture may need to be renewed. (Lindvall et al., 2004)

Gul et al. (2008) examined Extreme Programming in multiple companies in the telecommunications domain. For instance, Vodafone IT in Turkey and Nortel Netas in Turkey were involved. The finding was that Extreme Programming is to be tailored to fit the domain. The products are based on protocol stacks or applications built on the stacks. The standards are well defined. Thus, the learning is a time-consuming process. The complexity led to challenges in defining tasks from the items. The study results encourage utilizing collective code ownership, in-house coding standards, and customer presence as specified in the Extreme Programming model. In addition to the model, the study suggests preparing the top-level design documents and dividing the project into sub-modules. Also, the paper encourages adjusting the iteration length, not only within the projects, but also between the projects. (Gul et al., 2008)

The agile software development methods address to help in the challenges the telecommunications domain is facing. The volatility and uncertainty of the requirements as well as the knowledge sharing are acknowledged by the agile methods. In addition, the release scheduling is more sophisticated than in the traditional methods. Figure 19

visualizes the complexity management in Scrum. In the pre-game phase, the planning and prioritization takes place. The high-level architecture and the standards are recognized. The product backlog is utilized as a tool for steering the development as well as input to the sprint backlog, which in turn acts as an interface to the integration and system testing in the post-game phase. (Kettunen, 2009)

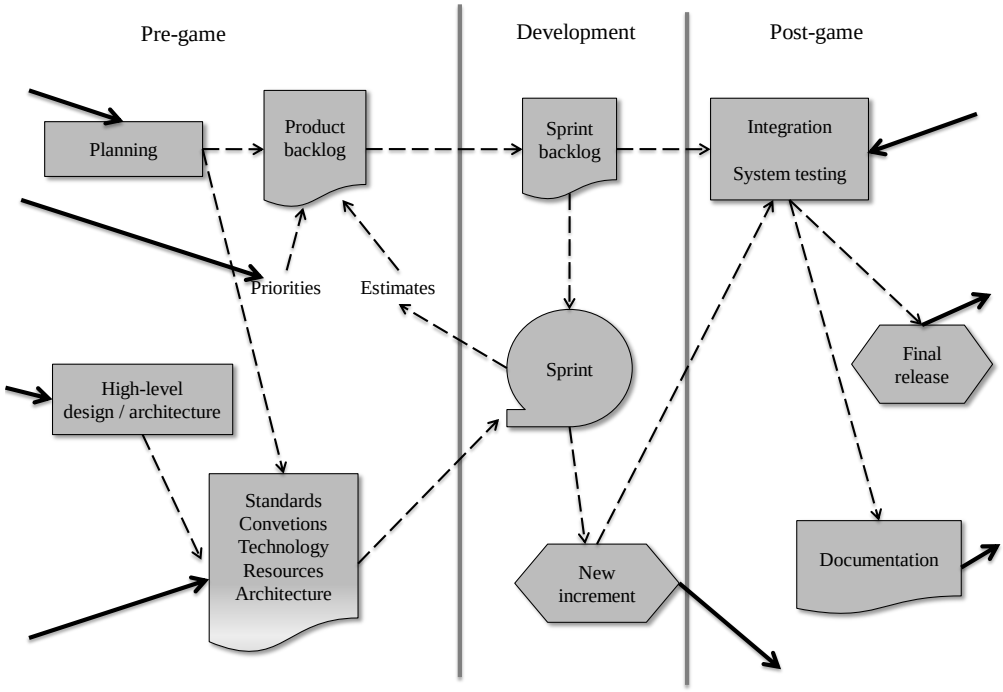

Figure 19. The organizational extensions with Scrum (Kettunen, 2009)

Kähkönen (2004) presents a concept of 'Communities of Practices' at Nokia. These communities are formed based on the informal relationships of employees sharing common practices. The communities exist as a complement to the formal team structure.

In addition Kähkönen (2004) presents three internally emerged agile software development methods at Nokia. The first method, RaPiD7, is a model of elaborating the lightweight documents of the new features in cross-functional workshops. The approach of seven steps proceeds from the preparation and kick-off to idea gathering and analyzing and even further to the detailed design, decisions, and closing. The concept involves the relevant stakeholders early. The second method, integration camp, combines the representatives of the teams to a one-day integration session. The third method, SEED, is a milestone model involving a representative from each team and the management of, for instance, requirements, architecture, integration, and releasing. (Kähkönen, 2004)

Also, from the product backlog perspective, the complex nature of the telecommunications domain is visible. Lindvall et al. (2004) identified that if a requirement is defined at a high level in the product development organization, the task of decomposing the item into the detailed specifications is demanding. At one unit of

Nokia, the product backlog was split into four levels. On the topmost level was a program content backlog that included all requirements of a started program. A rule was to mark from sixty to eighty percentages of the requirements as mandatory and the others as optional. On the second level, a program backlog for each release existed. The program backlog included all stories for the release and was managed by the program product owner. The program management measured the velocity from this level. The third level was the scrum team backlogs owned by the team product owners. The teams needed to utilize their own backlogs since the team might contribute to multiple projects simultaneously. The fourth level of the backlogs was the sprint backlog. (Laanti, 2008)

At one unit of Nokia, three high-level problem areas of scaling were identified. First, the transition from individual specialists to the cross-functional teams was identified to be challenging. Second, the management needed to change the role from the past, since for instance, prioritization was challenging in the same area previously managed. Third, the interest in developing the software development process might decrease once the agile model has been implemented. (Marchenko & Abrahamsson, 2008)

2.5.2 Agile in Network Equipment Vendors

The agile transition at Ericsson Netherlands enhanced the performance, transparency, and commitment. The cancelled requirements, i.e., aborted development, decreased from 25% to 9%, the time-to-market decreased by 50%, and the cost of maintenance by 20%. In addition, the meeting overhead was reduced. However, the ratio of fault-slip-through increased slightly, although, also design base faults, i.e., faults in the legacy code, were identified. (Goos & Melisse, 2008)

Once the scaling at Ericsson Netherlands was completed, communication and visibility increased. The scaled concept included a dedicated support team and a dedicated support backlog leading to delivering on time with high quality of service. The key success factors for the performance were the orientation to achieve, the continuous improvement, teamwork, focusing on growth, integration to the newest technology, and clear goals. (de la Rie, 2008)

At Ericsson Croatia, it was identified that the backward compatibility was required. The challenges arose regarding the coordination of the design capabilities, minimizing the effort needed for integration, and maintaining the level on quality of service. The solution was a milestone concept emphasizing tracking of the requirements and visibility. The items were prioritized prior to including those in the projects. The project was managed with a project anatomy mapping including a large amount of minor system enhancements, deltas. The incremental system growth enables testing the legacy code and new functions in parallel with the help of the weekly builds and high level of automated tests. (Ivček & Galinac, 2009)

Also Finland has contributed to the academic literature in terms of agile in the telecommunications domain. A combination of Extreme Programming and Pair Programming was experimented at Ericsson Turku, which was developing the Mobile

Media Gateway. The vision was to build competence and to increase the motivation of working. The developers experienced the planning game and splitting of the user stories into tasks as the most helpful tools. The number of written trouble reports decreased only by 5.5% in the pilot. However, the competence of the teams increased by 47%. (Auvinen et al., 2005)

Holmström Olsson (2009), describe the Streamline Development at Ericsson Sweden. The model emphasized continual deliveries, customer satisfaction through early deliveries, motivated individuals, face-to-face communication, simplicity, vision, and semi-permanent cross-functional teams. The teams included competence of design, test, and system management. (Holmström Olsson, 2009)

Nokia Networks (currently Nokia Siemens Networks) has been utilizing the agile software development models for multiple years. The employee satisfaction has been on a high level. Test Driven Development (TDD), Scrum, and other agile methods were emphasized through coaching and agile information. Materials, blogs, and information regarding events were, for example, distributed via wiki. Since the complexity nature of both the products and the organization, the Scrum-of-Scrums concept was utilized. (ITEA, 2006) Vodde (2006) recognized two mandatory concepts at Nokia Networks; short iterations and the retrospectives to provide quick feedback loops and enhanced adaptability.

The contribution of Nokia Networks to the research of the agile software development methods led to the Nokia test. The test was originally designed by Bas Vodde but afterwards enhanced by Jeff Sutherland. In the test, the organization answers to multiple-choice questions. The questions are related to, among others, the iteration length, the product owner, and the product backlog. Interestingly, the test provides more points if only one product backlog is utilized in comparison with involving multiple backlogs. In addition, the prioritization and usage of the product backlog as a tool for release planning are acknowledged. (Sutherland, 2008) Appendix B presents the Nokia test.

At Motorola, it was identified that a gap between the requirements and the teams exists. Thus, a concept of a bridge person was introduced in a tentative framework. The person attended the team meetings and provided lightweight documentation to create understanding regarding the requirements for the teams. In addition, the concept of a knowledge base was introduced. The base was supporting the growth of competency in addition to linking the agile and traditional methods. (Woi Hin, 2006)

In one unit at Motorola, the amount of escaped defects decreased as a consequence of implementing agile. In addition, the productivity improved. (Woi Hin, 2006) In another unit, Extreme Programming was piloted. The findings suggested that a roadmap was needed. Time needed for learning was reduced whereas the engineering productivity and the test coverage were significantly increased. However, one of the teams faced challenges due to not involving baseline architecture. The team needed to organize multiple re-factoring sessions. (Drobka et al., 2004)

At Alcatel-Lucent, a more-agile variation of the V-model was implemented. The weekly deliveries, minimized documentation, informal communication, cross-functional teams, job rotation, and joint planning were added to the V-model. However, the quality control was managed via gates. The point of accomplishing the product was called ‘general availability’, i.e., the point in which the product can be sold in the markets. (Leszak & Meier, 2007)

2.5.3 Agile in Operators

Also the mobile operators have implemented the agile software development models. The regulation and the dynamic competition shape the market continuously. The products involve third party components. At British Telecom, the transition process was long and painful. External help, the top management commitment and a critical mass of agile-minded employees were required for the transition. The company was able to shorten the development cycles and to enhance collaboration. One of the key success factors was to establish a clear business target for each sprint. The problems emerged for instance in the continuous integration since the components under development were shared across multiple programs. Thus, a system level architecture was still needed. (Evans, 2006)

Another problem at British Telecom was that the scaling spread the effort leading to a thin and thus ineffective effort. British Telecom promoted agile and provided an agile cookbook. The promoting included learning projects, agile road shows, agile program days and agile courses. The cookbook was a wiki-based solution to distributing information regarding the agile methods. The cookbook emphasized, e.g., the customer involvement, automated testing, iterative project tracking instead of milestones and discarding the Gantt charts. (Hanly et al., 2006)

At Telefonica, Scrum and Extreme Programming were introduced in three phases. First, a pilot including experimentation and acceptance was performed. Second, the agile means of working were launched including changes in the culture and in the organization. Also, a common understanding was codified. Third, agile was scaled in the enterprise rollout phase. The change addressed reducing the time-to-market, enhancing the business transparency and enhancing the innovativeness. The new means of working based on four pillars; the innovation initiatives, the knowledge groups, the strategic functions, and the common services. A new unit, the innovation and agile methodologies team, emerged. (Hornos & Izquierdo, 2009)

In the case of Telefonica R&D, three levels of backlogs were introduced. The Product backlog was on the highest level. The items that the teams chose for a sprint were moved into the product backlog for selected items, which included all the checked-out items. Simultaneously, the items were split into tasks and placed in the sprint backlog. (Hornos & Izquierdo, 2009)

To enhance the collaboration, the rooms were refurnished at Telefonica. In the new setup, the team members, the scrum masters, and the product owners sat around large

tables. The open space office included multiple whiteboards and inspiring corners for spontaneous meetings. The new setup also included sofas, a library, and private rooms. (Hornos & Izquierdo, 2009)

At T-mobile in the United Kingdom, Test Driven Development was introduced. The quality of the software increased as well as the cohesion of the code. The customer identification and the acceptance tests were identified as the key success factors. The developers emphasized the possibility of optimizing and adding features in a more flexible manner than earlier. (Rendell, 2008)

3 Research Methods

The literature review focused on introducing the Scrum framework, discussing metrics, innovations, and the state-gate model in agile, identifying options for scaling, and finally benchmarking companies in the telecommunications domain. In addition, the evolution of software development models was described.

This chapter describes the methods selected for the study. The interviews and the selected means of analyzing the interviews are presented. In addition, the chapter briefly discusses the context, i.e., the case company and the product.

The first step was to identify the potential stakeholders of the product backlog. From the literature review, the scrum roles were identified as stakeholders. In addition, potential stakeholders were identified from the open-ended interviews.

To maintain a clear structure on the empirical part of the thesis, an in-house framework for the new product development decision was selected as a framework for the research. To connect the stakeholders of the product backlog and the decision model, a matrix was formed. Open-ended and semi-structured interviews were needed to acquire the data for the matrix. According to Tengshe and Noble (2007), formulating an approval matrix is important from the corporate perspective to identify the authorized persons at each decision point.

The scope of the thesis is on the development of new functions, i.e., the customer requests regarding the already released products are discussed in a lightweight manner. Over the research, the key employees driving the agile transition regularly guided the study.

Figure 20 clarifies the structure of the research. With help of the literature review and the initial open-ended interviews (n = 6), the research questions were formed. Next, the semi-structured interviews (n = 11) were conducted. The decision framework was involved as a structure of the semi-structured interviews. The duration for both types of interviews varied between 20 minutes and one hour. The described matrix and the discussion regarding the needed data are the outcomes of the analysis. The outcomes are interconnected.

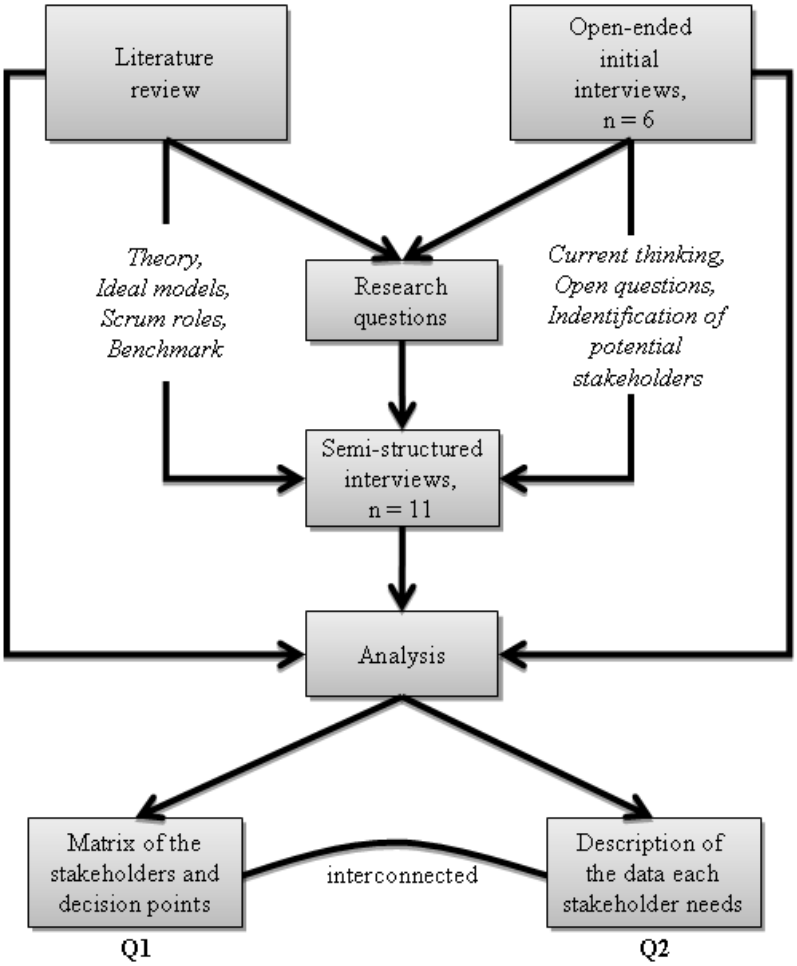

Figure 20. Structure of the research

3.1 Interviews

The initial step, prior to choosing methods and research questions, was to conduct open-ended interviews. In total, six interviews, including one interview of two people simultaneously, were conducted between December 2009 and February 2010. Literature was reviewed in the same period. The interviewees were working with the pilot scrum teams, managing the transition phase, or developing the high-level innovation framework for Mobile Media Gateway development at Ericsson Finland. Due to the privacy and confidentiality issues, the names or job titles are not mentioned in the thesis.

The semi-structured interviews began by a discussion regarding the role of the interviewee in the new agile means of working. After the short discussion, the decision framework was discussed step-by-step from the perspective of the interviewed stakeholder. Depending on the flow of the interview, the decision points were discussed verbally or with the help of a print of the blank matrix with only headers. The matrix included only the suggested stakeholders in the vertical axis and the decision points in

the horizontal axis. Hence, the horizontal axis was a timeline for the framework. The cells, to which the data was gathered in the interviews, were left initially blank. Thus, the interviewees could present their thoughts and experiences without the interference of the data gathered from earlier interviews. Appendix C presents the matrix involved in the interviews.

The interviewees for the semi-structured interviews included department managers, experts of different areas involved in the agile transformation, and people acting in the scrum roles. Tables 4 and 5 list the interviewees and the dates of interviews.

Table 4. Open-ended interviews

The Interviewees	**Date**
A manager in the agile transition	23.11.2010
	25.2.2010
A proxy product owner (PPO)	5.2.2010
A scrum master (SM)	9.2.2010
A scrum team member	25.2.2010
Innovation managers (two interviewed simultaneously)	2.3.2010
Total	**six interviews**

Table 5. Semi-structured interviews

The Interviewee	**Date**
Department managers	3.3.2010
	17.3.2010
	31.3.2010
	1.4.2010
An Early Phase Program (EPP) expert	8.3.2010
A scrum team member	22.3.2010
A scrum master (SM)	25.3.2010
Proxy product owners (PPO)	12.3.2010
	19.3.2010
A post-GA activity expert	1.4.2010
A release verification expert	31.3.2010
Total	**eleven interviews**

3.2 Analysis

The analysis began by combining the findings from the interview notes. The findings were grouped per stakeholder. The findings are discussed stakeholder by stakeholder in Chapter 4. Next, the matrix was completed based on the findings. To enhance the utility of the matrix, only the major findings were stated. The findings from literature were compared with the findings from the interviews. The differences and the similarities were mapped to the matrix. Based on the matrix, the decision process can be enhanced to utilize best practices from literature.

Conclusions from the matrix were interpreted to provide a solution for the research questions. In addition, means of visualizing the product backlog direct and indirect data were created and identified from literature.

3.3 The Research Context

The research is conducted at Ericsson Finland R&D, located at Jorvas, Kirkkonummi. The unit is developing the Mobile Media Gateway (M-MGw), which is part of the Ericsson Mobile Softswitch (MSS) product portfolio. The gateway utilizes a platform. Both hardware and software are developed for the gateway. The study focuses only on the context of software development. In addition, the study was limited to the development of the Mobile Media Gateway, i.e., the product portfolio level was discarded.

The development of the gateway employs hundreds of engineers. The engineers are located in Finland, Hungary and Germany. Previously, the software development process has been a variation of the V-model. At the beginning of the research, the Scrum framework was experimented by one trial team in Hungary and by two trial teams in Finland. The decision for scaling the Scrum framework to the large extent had already been formed prior to the thesis. In addition, the decision framework had already been planned.

3.4 Alternative Research Methods

The pilot teams had been established for approximately three months. There was no feasible quantitative data generated yet. Thus, the mathematical approaches were obsolete.

Qualitative research methods include, among others, observation, questionnaire, focus group, and interviews (Silverman, 2006). Since the thesis address forming suggestions to the scaled Scrum implementation, the observation of the stakeholders involved in the pilot would not have provided useful knowledge. The questionnaire was not utilized since the direct questions were impossible to form due to the earliness of the Scrum implementation.

The focus groups were not utilized since most of the stakeholder roles were handled by only a few employees. In addition, the interviewees were extremely busy. Hence, it would have been challenging to arrange a meeting.

Since the other qualitative methods were blocked, interviews were selected as the method for the research in addition to the literature review. As stated, no direct questions could have been formed. Hence, the options were to utilize open-ended or semi-structured interviews (Silverman, 2006). The open-ended interviews were selected for identifying the stakeholders and the research questions. However, semi-structured interviews were selected for gathering the data regarding each stakeholder at each decision point.

Qualitative data is subjective, speculative, and flexible (Silverman, 2006). Hence, the data gathered from the interviews represent the interpretation of single actors. Indeed, the data is subjective and prone to be biased by personal interests. In addition, it is important to note that the transition was in an early phase, i.e., the interview results are partly guesses or faulty assumptions by the interviewees. Also, the data is subjective and speculative in terms of the researcher.

Due to the open-ended and semi-structured nature of the interviews, the results are not likely biased by the interviewee dictation of the conversation. However, structured questions would have revealed more comparable data. As stated, due to the earliness of the transition, the direct questions were impossible to form.

Interview as a method was feasible for the research in the current stage of the Scrum implementation. Since the ultimate stakeholder activities were not decided, it was proper to select a method that enables discussion and additional questions.

4 The Stakeholders and Decision Points

The chapter defines the stakeholders and their activities in different phases of the decision process. According to the research questions, the focus is on the product backlog. The chapter analyses the semi-structured interviews by reflecting the literature review. Prior to introducing the findings regarding the stakeholders, the new product development decision framework is introduced since it is utilized to structure the findings. In addition, each of the stakeholders is discussed one by one. Chapter 6 summarizes the findings.

The stakeholders of the product backlog were identified from two sources. First, literature discusses the scrum roles that obviously are stakeholders. Second, the initial open-ended interviews supported the identification of the stakeholder. The managers above the product owner team, e.g., the product management and the manager of the R&D, are utilizing the product backlog only through the product owner team. Hence, these managers were not stated as the stakeholders of the product backlog.

4.1 New Product Development Decision Framework

Over the pilot of Scrum, a new product development decision framework has been formed in-house. The framework includes five decision- and checkpoints, marked with the letter F. The first stage is ‘F0’ and the last stage intuitively ‘F4’. The framework emphasizes the release management through the checkpoints and agility through the team contribution and commitment. Thus, the framework bundles the state-gate model (Cooper, 2000) and the Scrum framework (Schwaber, 1996) as discussed in Chapter 2.3. The combination was identified to be a possible solution for instance by Karlström and Runeson (2005) but discouraged by Larman and Vodde (2009). It is important to note that the study discusses the decision framework on the Mobile Media Gateway level. The framework is different at the MSS portfolio level.

At F0, a group of solution area experts, the Early Phase Program (EPP), is requested to create a lightweight one-page document of the new feature or improvement. It is a new team introduced by the new decision framework. The customer requests are input to the group. According to the agile manifesto (Fowler & Highsmith, 2001), it is acceptable to receive changes to the customer requests. However, according to the decision framework, the customer requests are first handled in the product management. In future, the plan is to utilize the in-house ideas also as input.

Currently, the innovation process is not integrated into the EPP in such a manner that the ideas would flow smoothly from the idea management and collaboration tools for the EPP. The decision framework suggests that all of the new features and improvements are routed via the EPP program if there is no existing one-pager to which the change could be bundled. Thus, a finding from one of the interviews was that an inefficient Early Phase Program increases the length of the feedback loops due to the routing. If the length of the loops increases, waste (Benefield, 2008; Poppendieck, 2002) is generated in form of delays.

The EPP team performs early investigation and writes a one-page document of the new feature or improvement between F0 and F1. The paper includes, among others, a short description, an identification of dependencies to the third parties and to the platforms, and the expected environment changes. The workload of writing a paper is tens of hours. The one-pagers are not placed in the product backlog as such. Instead, they are placed in a separated list of one-pagers. However, there were no in-house arguments supporting or not supporting the avoidance of the product backlog in the handling of the one-pager. According to literature (e.g., Racheva & Daneva, 2008), the items can be placed in the product backlog without immediate commitment to implement the item. In addition, the items can be prioritized based on business value and risk (Barton, 2009), i.e., if the one-pagers were placed in the product backlog, they could be initially large items with uncertain business value and low priority. Over the time, the items would be split and the uncertainty would be minimized. Also, the Nokia test (Sutherland, 2008) encourages utilizing only one product backlog. A separated list of one-pagers is similar to a separated backlog.

Since the Mobile Media Gateway, is part of the Mobile Softswitching product portfolio, one interviewee raised the question whether the one-pagers are to be mapped to the portfolio level. In addition, the same interviewee believed that it might be feasible to prioritize the one-pagers.

Currently, the Early Phase Program team does not include representatives from the scrum teams. The investigation is conducted at a high level. Thus, it is suggested that the scrum team members contribute later. However, the paper should identify dependencies and provide insights for the needed environment changes that might need the expertise of the scrum teams. In addition, the EPP team does not include product owner team members unless requested.

Overall, the Early Phase Program is similar to the opportunity team in the case study by Nahor and Katzav (2009). In their study, the case company utilized an opportunity team that evaluated and linked ideas and finally placed the potential ones in the product backlog.

At F1, the feature is placed in the product backlog by the product owner team. Simultaneously, a handover of responsibility is performed from the Early Phase Program to the product owner team. However, the handovers violate the lean principles (Benefield, 2008; Poppendieck, 2002). While inserting the item into the product backlog, a priority compared with the existing items in the product backlog is to be stated.

Between F1 and F2, further investigations are performed. The output is a feature concept study (FCS), initial user stories, a verification analysis, and cost estimates. The FCS includes technical knowledge such as dependencies and architectural solutions. In addition, the implementation aspects can be discussed with, for instance, the developers. It can be argued whether the FCS is aligned with the principle “Just enough, just-in-time specifications” by the Nokia test (Sutherland, 2008) since the FCS includes relatively detailed technical definitions. According to the interviews, the desired behavior is to

conduct only light-weight FCS. However, in reality, the FCS often describes the technical solutions on a too detailed level.

At F2, the investment decision is formed by the product management, i.e., the feature will be implemented or discarded. Also, representatives from, e.g., marketing and services are involved in forming the decision. In addition, the MSS portfolio level is involved if the new feature impacts other products in the portfolio. Inputs to the decision are, e.g., costs, the available resources, the schedule, and an estimate for net present value. At F2, there is still possibility to change the priorities. The implementation is performed between F2 and F4.

Between F2 and F3, the user stories are selected to sprint in the sprint planning. In addition, prior to F3, all of the user stories are split into proper size. The scrum teams perform the highest priority stories and are mandated to process changes within the limits of the previous feature definitions. The costs are tracked continuously. The priority of a feature can still be decreased.

At F3, a commitment to the feature release date is stated. However, the commitment is conditional – if the priority is changed or the velocity is notably changed from the estimated, the commitment becomes obsolete. Between F3 and F4, the priority of the item cannot be decreased in the ideal framework. The scrum teams complete the coding, function verification, and partly the system verification. Function verification is related to testing the functionality of a single feature whereas system verification is related to testing the feature integrated into the product.

At F4, the feature is implemented. It is integrated and tested except final release verification. A handover of responsibility is conducted from the scrum teams and from the product owner team to the release project. The release projects need to be prepared for the handover. For instance, the verification analysis and environment configurations are to be ready prior to the handover to eliminate the idle time, i.e., waste.

The outcome of the process is a minimum marketable feature (MMF), i.e., the feature is released to market early including possibly only the most critical functionality based on the market demand. The less-critical parts can be added to the next releases. Thus, with the concept of MMF, the time-to-market of a new feature is decreased. The concept is aligned with the lean principles; nothing extra is added to the feature (Benefield, 2008; Poppendieck, 2002) and the feature is pulled by the customer demand (Raman, 1998).

At the same time with the implementation, parallel independent testing is performed. In addition, the integration and the release verification are performed outside the scrum teams. Thus, an agreement for dividing the verification tasks is needed. Along the path from F0 to F4, the feature portfolio status, the release status, and plans regarding the external release are monitored. In addition, the roadmaps and the priorities are to be aligned throughout the development process.

Once the release project ends, the release gains the general availability (GA), i.e., it can be sold and delivered to the customers. Figure 21 visualizes the new product development decision framework.

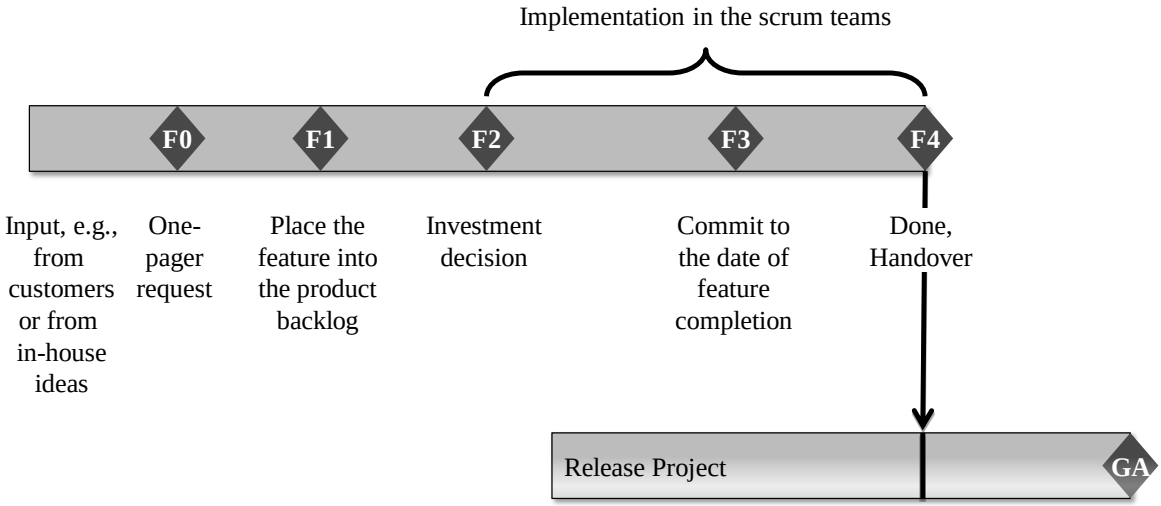

Figure 21. The decision framework

4.2 The Innovation Process

In 2009, in the case company, enhancing the innovativeness was in a special focus. The managers of the innovation process introduced new concepts such as the innovation day, the innovation coaches, and the new idea gathering tool in the intranet. In the interviews, it was identified that the link between the innovation process and the Scrum framework is currently mild. However, the interviewees agreed that the innovation process is an important function of the new product development.

Hornos and Izquierdo (2009) describe a team that enhances the innovativeness and agile means of working. By combining the innovation process managers and the agile methodology managers, synergies and integration are expected. However, as stated, in the case company of this thesis, innovativeness and agile are managed separately. To emphasize the importance of the innovativeness, the innovation process was selected for the scope of the study as one of the stakeholders for the product backlog.

The suggestion by the innovation process managers was that there would be a separated innovation backlog in the scaled Scrum implementation. It was thought that if the new ideas involving high uncertainty were placed in the product backlog as such, they would not be prioritized higher than the items planned to be implemented. In addition, it was suggested that the scrum teams are mandated to spend a fixed portion of time only for the items of the innovation backlog.

The situation is partly similar to the EPP. Both the EPP and the innovation process involve a separated backlog. However, Racheva and Daneva (2008) encourage placing

all, including the uncertain, items in the product backlog. Also, the Nokia test (Sutherland, 2008) clearly discourages multiple product backlogs.

According to a suggestion by the innovation process managers, the items from the innovation backlog would be investigated and split by the scrum teams that have dedicated time for the items of the innovation backlog. Once the investigation and split are conducted, the items are to be placed in the product backlog. However, the interviewees were not able to offer a solution for linking the suggested innovation backlog and the product backlog, i.e., how to transfer the items from the innovation backlog to the product backlog.

Another stakeholder stated that the situation is opposite. If the items were placed in an innovation backlog, the scrum teams likely select items only from the product backlog. This is due two reasons. First, the product owner team emphasizes the importance of the items of the product backlog. Second, the items in two different backlogs cannot be compared in terms of priority. Thus, the scrum team is not aware whether to select items from the product backlog or from the innovation backlog. In this case, the scrum teams probably select items from the product backlog since the items are aligned with the market demand.

4.3 The Product Owner Team

The product owner team is responsible for keeping the product backlog prioritized. The team consists of a product owner (PO) and multiple proxy product owners (PPO). The proxy product owners are jointly responsible for the features. The concept of PPO is similar to the area product owner concept by Larman and Vodde (2009). However, the area product owners are strictly limited to a specific area of features. As opposite, the proxy product owners share the responsibility to eliminate the handovers due to, e.g., vacations and other absences.

The product owner team is an interface between the product management and the scrum teams. In short, the product owner tasks can be divided into two; secure that the scrum teams complete the proper items and secure that the scrum teams have enough work to do. The scrum teams might provide input to the split but the product owner team is solely mandated to modify the product backlog items. The product owner team is responsible for the prioritization. In addition, the product owner team is responsible for the budget. The team presents the current status of the development and the budget to the product management.

At F1, the Early Phase Program has completed the one-pager and deeper investigation. The product owner team receives the responsibility of the feature. Between F1 and F2, a feature concept study is written by the product owner team. The initial concept was to select one proxy product owner as responsible for each FCS. However, the current solution is to collectively share the responsibility between the product owner team.

The FCS states the dependencies and partly also the technical solution. In addition, the analysis of the test scope is mentioned. In the FCS, the mandatory and optional

functionalities of the feature are stated, i.e., the time-to-market can be reduced by not implementing the optional parts into the first release according to the minimum marketable feature principle. As stated earlier, the alignment of the FCS and the Nokia test (Sutherland, 2008) can be argued.

At F2, the feature concept study is completed. The product backlog items are split by the product owner team in such a manner that the scrum teams are able to select the first items post the F2 decision and thus can start immediately to implement the feature. By the F2 decision point, the business aspects are examined to provide the needed data for the product management to form the investment decision. In addition to the revenue estimates, lead time, interfaces, and dependencies are known at F2. All uncertainties postpone the F2 decision. The data for the decision is primarily provided by the product owner team.

Since a proxy product owner is responsible for the feature concept study, the scrum teams have limited knowledge of the new feature at F2, i.e., at the point to start the implementation. Thus, a partial handover between the product owner team and the scrum team is conducted. As stated, the handovers violate lean (Benefield, 2008; Poppendieck, 2002). However, since the proxy product owners continue to support the scrum team, there is no pure handover. It is to be noted that the current organizational thinking does not equal the ultimate solution. A suggestion by one interviewee was that the scrum teams would be involved in the feature concept study in future.

Between F2 and F4, the product owner team calculates the velocity for the feature, i.e., the sum of velocities of the teams. Based on the collective velocity, the product owner team can calculate an estimate for the date of feature completion. The release management is interested in the date since it is utilized in deciding whether the feature fits the next release or not. Since, from the product owner team perspective, the velocity is utilized for estimating the duration of development, according to the interviews, the velocity of a single team is not in the interests of the product owner team.

At F3, the proxy product owners evaluate the state of the timetable, the technological issues, and the budget. In addition, the amount of risk is studied. The risk is related to the number of trouble reports, the number of impediments, and the effective time spent on implementing the feature. At F3, the proxy product owner states the date of feature completion, i.e., the F4 decision point. However, two boundary rules are set; the velocity and priorities are to be maintained on the estimated level. If the boundary rules are discarded, so is the estimate for the date of feature completion.

In addition, at F3, an estimate for the contents of the feature is proposed along with a list of optional parts that are implemented if the teams perform above the estimations. Thus, the minimum marketable feature concept dynamically shapes the scope of the new feature implementation.

At F4, the product owner team hands over the feature if the quality is on an acceptable level. The product owner team is responsible for accepting the quality. The handover is performed to the release management, i.e., the feature is integrated into the release and

the release verification is executed. If the feature is to be delivered also to the releases in the field, the post-GA activities, i.e., the customer support, receives the feature with a handover from the new feature development.

4.4 The Scrum Masters

In the case company, the abbreviation for scrum master is 'ScM' since the abbreviation in literature, 'SM', stands for section manager in the case company. The academic abbreviation SM is involved in this thesis.

The interviewees emphasized the role of the scrum master to be a coach. The scrum masters are not mandated to form decisions at F-points nor does have the mandate to form decisions regarding the sprint. At least in the scaling phase, the scrum masters are coaching all stakeholders identified in the matrix in Appendix C. The coaching role of the scrum masters towards the whole organization was suggested by Schwaber (2009). In addition to coaching, the scrum masters enhance the transparency of the product development.

Over the EPP program, i.e., until F1, the scrum masters are coaching the EPP team and attending the EPP meetings for a few hours a week. Between F1 and F2, the feature concept study is written by the product owner team. The scrum masters are also coaching the product owner team. Thus, they are involved in the feature concept study in the role of mentors. One interviewee suggested that in the future, the scrum masters could organize user story writing workshops for the scrum teams at F1 or F2. This would help the feature concept study writing process and involve the scrum team earlier than in the current solution. Thus, it would remove the identified partial handover at F2, i.e., the partial handover from the product owner team to the scrum teams.

At F2, the scrum masters start to mentor the scrum teams that are completing the items. The task of the scrum masters is to remove impediments and to enhance the software development process within the teams. To keep the sprint planning meetings short and to pre-identify dependencies and impediments, a product backlog grooming session is held on every other week. Over the meeting, the teams discuss and split the items to be ready for the next sprint planning meeting. The scrum masters organize the product backlog grooming sessions. The role of facilitating and removing the impediments is aligned with Eskelinen et al. (2010).

Overall, according to the interviews, the scrum masters are not interested in the contents of the product backlog – they are interested in the means of exploiting the product backlog. However, the scrum masters are interested in calculating the velocity and in measuring the changes of the product backlog. The velocity calculations reveal the performance of the teams. Thus, it is in the interests of the scrum masters and the teams to be able to enhance the process. Tracing the changes in the product backlog can be utilized in two aspects. First, it reveals the progress, i.e., the current performance. Second, it reveals the amount of new items the product owner team has added to the product backlog.

4.5 The Scrum Teams

The term for a scrum team in the case company is ‘a cross-functional team’ (xFT). In this thesis, the academic term ‘the scrum team’ is involved. Between F2 and F4, the team performs user stories to complete a feature. One interviewee suggested that the team should be involved earlier, e.g., in the Early Phase Program or in the feature concept study, to define the needed changes to the test environments and the tools. In addition, this would remove the partial handover from the product owner team to the scrum teams discussed earlier.

Inputs to the scrum teams are the feature concept study and the initial user stories. Over the sprint planning meeting, the team selects items from the product backlog. However, the product owner team provides suggestions for the selection. In addition, the prioritization is conducted by the product owner team. The ultimate selection of items is decided by the team. Thus, the self-organization of the scrum teams is enhanced (Schwaber, 2009). Besides the items related to the product, the changes to the tools can be managed as product backlog items.

In every other week, a product backlog grooming session is held. The concept of the grooming sessions is aligned with the Scrum framework (Schwaber, 2009). In the session, the team discusses the items in the product backlog that are not yet checked out. More items are under discussion than the team could select for the next sprint to be prepared also for the coming sprints. In the grooming session, teams may decide to suggest a split of items for the proxy product owner who is the only authority to split. According to Kniber (2007), the proper sized product backlog items are a key success factor.

The main interactions between the scrum teams and the product backlog can be divided into three categories; the sprint planning, the product backlog grooming, and utilizing the product backlog as a gateway to information. The last category involves actions such as utilizing the product backlog as a link to the wiki, to the definitions, or to the list of contacts. Also Deemer et al. (2008) suggests that the product backlog should include links to the wiki, which includes more detailed description. In addition, the product backlog is a useful tool for interpreting a wide perspective of the product. Hence, multiple views to the product backlog are needed.

As stated by Nahor & Katzav (2009), the product backlog should involve different views for different stakeholders. In the case company, the pilot scrum teams involve their own views in the product backlog. The view includes for instance the items the team has selected for the current sprint and the items completed earlier. In addition, the view reveals the product backlog items that have been marked for the team. The pre-selection of the teams to complete the items is needed for instance if the item requires specific competence.

The sprint backlog is maintained only on the whiteboard, i.e., there is no electronic link between the product backlog and the tasks in the sprint backlog. Thus, the product

owner team is aware of the status only at the end of the sprint, not over the sprint. Hence, the cycle for the data is two weeks.

Majority of information, for instance, the feature concept studies, is stored in a wiki. In addition, reports by the scrum teams and the description of the verification are stored in a wiki. In some cases, it is necessary to be aware of details regarding the earlier testing phases. Hence, the wiki-tool needs to include version tracking.

Earlier the testers have written a trouble report (TR) of each identified fault. In the Scrum framework, the team writes trouble reports only if the fault is identified in a code that is created by others or is done by the team itself but has already been internally released. If the code is not yet released outside the team, a trouble report will not be written. If another team identifies the fault, a TR is written. In this case, the product owner team places the trouble report in the product backlog and prioritizes it compared with the other items in the backlog. If a fault is identified in the code once the product has been released to the customer, the TR will be placed in another backlog discussed in Chapter 4.7.

Since the scrum teams test simultaneously with the coding, plenty of faults are identified within the teams. However, the faults are fixed within the sprint. The team delivers only high-quality code according to the definition of done. An exception to the definition is an agreement between the release verification and the scrum team that might postpone some testing activities to the release verification phase. This is feasible, for instance, if the scrum team does not have the proper test environment. Additionally, the regression tests can be launched automatically with, e.g., two hour interval.

4.6 The Release Verification

The release verification is not executed in the scrum teams. The concept of Scrum-external system verification is aligned with the findings of Kniber (2007). The release verification can be divided into two categories; prior-to-F4 verification and post-F4 verification. Figure 22 visualizes the categorization.

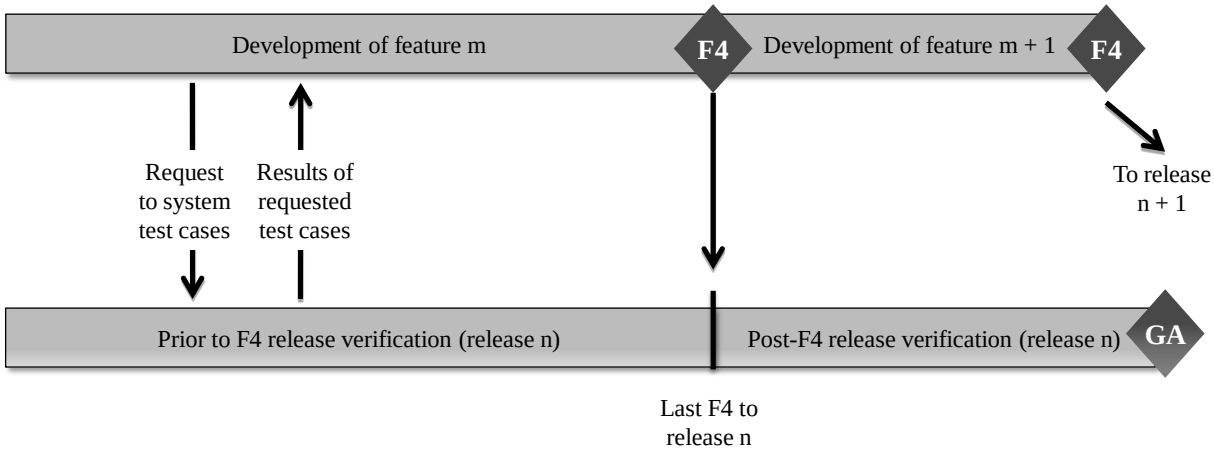

Figure 22. The release verification

The scrum teams implement the feature between F2 and F4. Over the period, the early system testing is executed. However, the scrum teams might not involve the proper test environment. Thus, the proxy product owner requests support from the release verification testers for a portion of the system test. The testers execute the requested test cases. Hence, a pull approach is utilized in the support for the scrum teams. To gain the proper knowledge regarding the implementation of the feature, one tester performing the release verification participates in the sprint review demonstration sessions.

At F4, the product owner team is responsible for accepting the quality of the feature. An internal release, i.e., a handover to the release project and possibly to the post-GA activities is performed. However, the criteria for the F4 decision are to be concluded in advance, preferably at the latest on F3. The criteria are to be accepted both by the scrum teams and the release verification testers. It is to be agreed which portion of the system testing is performed by the scrum teams and which by the release verification testers.

Once the handover of the feature is conducted from the scrum teams, the release verification is executed as a part of the release project. Hence, in addition to the date of F4 decision, also the release verification affects the readiness of the feature, i.e., it affects which product release the feature can ultimately be marketed.

To be prepared for the handover at F4, the release verification experts are to be involved in the feature specification process. If the changes in the tools and environment are not performed in advance, the lead time increases, i.e., waste is generated and the fit to the current release is compromised. There are multiple options for involving the release verification testers early in the feature development. First, the release verification testers provide expertise for the EPP prior to F1 if requested. Second, the release verification testers can be involved in the verification analysis between F1 and F2. Third, the release verification testers can be involved in designing the items to be placed in the product backlog. A release verification expert suggested that a brainstorming session is to be held for discussing the future items of the product backlog. In the meeting, each party would state their opinion and future actions.

An interesting opinion from a release verification expert was that also the release verification phase could utilize a backlog. Currently, the selection of the test cases is fixed, i.e., once a test case has been selected, it cannot be deselected without proper arguments. However, if a dynamic backlog would be involved, the release verification experts could select a large amount of test cases and sort those based on the priority. The commitment to execute would not be required for all items in the backlog. The priority could change over the verification. Hence, the low priority items could become high priority items, and vice versa, depending on the results of the test cases. With the suggested approach, it would be acceptable not to execute the lowest priority test cases as long as the highest priority test cases are executed properly. With the current concept of a fixed selection, all of the selected test cases are to be executed prior to GA.

4.7 The Release Management and the Post-GA Activities

Once a feature is ready, it is integrated into the product and a handover to the release project is conducted at F4. An indication of the date is stated at F3 by the product owner team. In addition, the minimum content of the feature is stated. From the release management perspective, it is essential to know the date of feature completion in advance to be prepared and to decide the scope of the release. In addition, the release project is interested in the quality of the code it is receiving, the possible unfixed faults, and the overall knowledge of the feature.

Figure 23 illustrates the release project management. The product is released to the market at point of general availability. Also Alcatel-Lucent utilized the concept of the GA stage (Leszak & Meier, 2007). Whether the post-GA activities are part of the release management or not is a matter of definition. In this thesis, the release management and the post-GA activities are discussed separately. The activities include reacting to the faults identified in the released products and reacting to the customer requests. The support of the released version is offered to a long time period, i.e., the new versions will be released while the old versions are still supported. A mapping between versions and backward-compatibility is required. If a fault is identified post-GA, a trouble report is written. According to the current thinking in the organization, separated post-GA backlogs are needed for the trouble reports and for the customer requests. The reasons for implementing separated backlogs include the differences in the working methods. The new feature development applies Scrum whereas the post-GA activities apply Kanban. In addition, according to the interviews, it is not sure whether the scrum teams and the post-GA teams share a common vision of the backlog on the conceptual level.

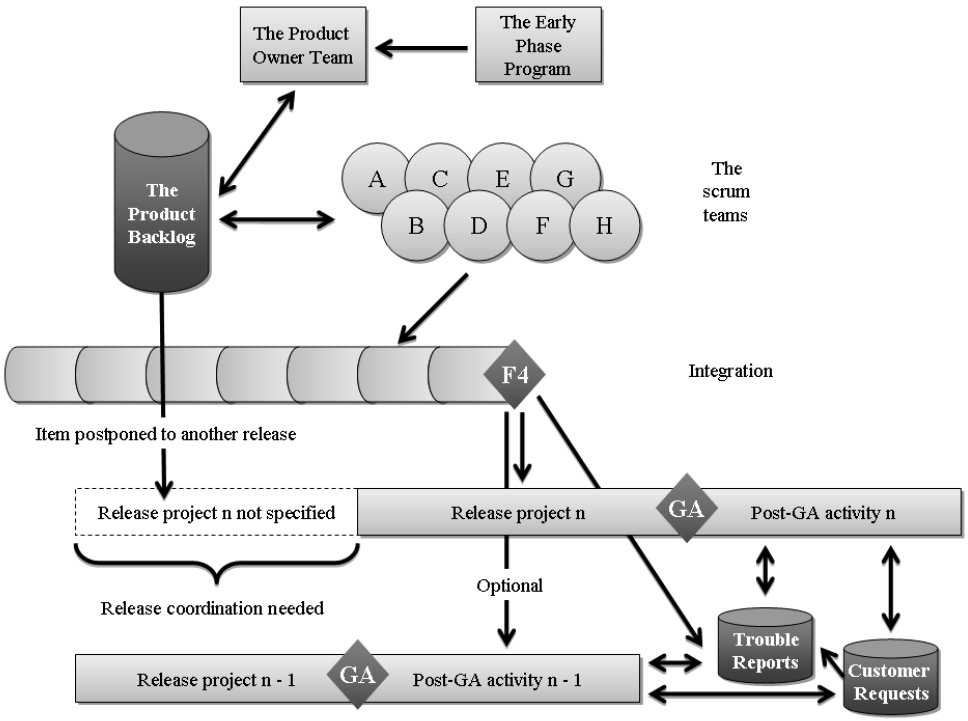

Figure 23. The release management

The trouble report backlog and the customer request backlog are handled by support teams specified only on one of these two areas. Interestingly, the customer requests may trigger trouble reports that need to be responded by the fault handling teams. Hence, there is a link between these two backlogs. It is to be noted that the support teams rely on physical whiteboards instead of software tools in the backlog handling. Again, the separated backlogs violate the desired behavior of the Nokia test (Sutherland, 2008). However, the concept of utilizing a separated support team was identified as a key success factor at Ericsson Netherlands (de la Rie, 2008).

Important conclusions can be drawn from Figure 23. First, if an item is to be postponed, it might be that the new release project does not yet exist. Thus, release coordination is needed to secure that the item is in the scope of the release project n. Second, at F4, the handover of a new feature might occur towards the post-GA activity in addition to the current release project. Hence, sales can be enhanced by offering the new feature to the addressable infrastructure, i.e., to the previously sold gateways in addition to the new gateways. It is important to note that Figure 23 is a simplification of the real world. In reality, at F4, post-GA activities for multiple previous releases are often ongoing. Third, the post-GA backlogs for the trouble reports and customer requests are common for all releases still supported. This implies that it is to be decided to which release the fix or improvement is to be delivered. Previously, the customer requests have been assigned as soon as they have been analyzed. In the new concept, the requests are placed in a backlog. Hence, there is no initial statement of the date of the response to the request or fault. The new solution of operating without a fixed date requires a change in the mindset since currently a statement for the date is required.

It is to be noted that currently the release project and the post-GA activities are not totally integrated for two reasons. First, the organization structure and the means of working at Ericsson Finland are not in a stage that the release project and the post-GA activities could be merged. Second, at the MSS product portfolio level, the desired behavior is to separate these to functions. Thus, a handover is conducted at GA.

In future, the plan is to rotate the developers between the new feature development and the support teams. In addition, it is yet under discussion whether the support teams and the new feature development teams could involve a common backlog for the trouble reports. According to one interviewee, if the backlogs are separated, there is a risk that the fixes are to be completed both in the scrum teams and in the support teams. Duplicate work demands extra effort that is obviously waste (Benefield, 2008; Poppendieck, 2002).

The contribution of the release project and the post-GA activities is minor prior to F3. If the new feature does not impact the hardware, the post-GA activities are involved in the discussions between F3 and F4. However, involving the support personnel late in the new feature development generates knowledge gaps. The post-GA developers can setup the environment rapidly but acquiring the needed knowledge is a time-consuming process.

Three interesting dependencies were pointed out in the interviews. First, as stated, some items may be postponed. The desired handling of these items was unspecified in the period of the research. The problem is that there might be no release project yet to receive the items. Second, the customer requests and the trouble reports regarding the products in the field are handled by the support teams. However, the expertise of the scrum teams might be needed in order to solve the issues. Third, the scrum team performs the function test and the early system test. However, the team members can provide insights of the most critical parts of the function, i.e., they might provide insights of functionalities that should be in a special focus in the release verification. Thus, the scrum teams and the last testing activities need to share a feasible communication channel that is both flexible and reliable.

Multiple benefits of the Scrum framework from the release management perspective were identified in the interviews. Scrum enhances transparency, predictability, and self-organization while decreases lead times. As an outcome, fewer actions for adjusting are needed in the release management phase. Both the release project and the post-GA activities are mainly interested in the date of feature completion, quality of the feature, and the scope of the feature. The concept for the scope is the minimum marketable feature. The release management interest in the date of feature completion was also identified by Sulaiman et al. (2006).

5 Product Backlog Direct and Indirect Data

As discussed in Chapter 4, some of the stakeholders utilize data for estimating and follow-up, some for implementing the new features. This chapter discusses the most important types of data that the product backlog directly and indirectly provides. To enhance the feasibility of data, it is to be visualized properly to meet the demands of each stakeholder. The transparency is to be emphasized. Hence, criteria for data are the manner it is visualized.

The data can be utilized in measurement purposes. Racheva and Daneva (2008) categorized the metrics into measurements at the code level, productivity metrics, and economic metrics. This chapter discusses the data that can be utilized in economic and productivity metrics. The code level measurements are not discussed, since the Scrum framework does not state which coding practices to involve. If the data is utilized for measuring, it is important to predefine the purpose of measuring. For instance, measuring can address enhancing throughput, velocity, and quality or decreasing lead time and headcount.

The chapter discusses velocity, estimate for work to be done, dependencies, and business value based on the semi-structured interviews. The chapter defines the stakeholders for each data. With the help of the definition, suggestions are provided regarding means for visualizing the data for different stakeholders. Chapter 6 summarizes the suggested combination, i.e., a framework for, visualizations.

5.1 Velocity

The concept of velocity has been discussed in Chapter 2.2.10. The velocity indicates the number of user stories a scrum team performs. The velocity data is utilized, for instance, as input to the release planning and to improve the operational excellence of the teams. The velocity deviates between the sprints and between the teams. The attributes affecting the velocity include, e.g., volume and dependencies. Volume is a function of the number of the scrum teams, the number of the product owners, the number of the product backlog items, the size of the product backlog items, and the number of the backlogs.

Based on the interviews, it can be concluded that the velocity is to be measured on multiple levels. The main difference is that only the team level velocity indicates the velocity of a single team. Hence, only the team and the scrum master are allowed to calculate the figures. The team level velocity can be divided into two aspects. The first aspect is the velocity within a sprint. The velocity is updated daily on the whiteboard, i.e., the cycle is 24 hours. It is illustrated with the help of the sprint burndown chart. Thus, it is measured in tasks. The burndown chart is an internal manner of the scrum team. As discussed in Chapter 2.2.4, the sprint burndown chart reveals warning signs. In addition, the burndown chart can be utilized as input to the sprint retrospective to discuss the progress.

The second team level aspect is the velocity over the sprints, i.e., the cycle is two weeks. The velocity of multiple sprints can be utilized as input to the sprint planning as well as to the retrospective meeting. The velocity over sprints reveals the enhancements in the means of working. The longer the time period in the scope, the more accurate data is since the effects of deviation are minimized. In addition to tracking minimum, maximum, and current sprint velocity, there are also other options for interpreting the data. For instance, average, weighted average, or the current trend can be calculated. The accuracy of the extrapolation is enhanced by selecting more sprints for the scope.

Figure 24 illustrates the team level velocity visualization over the sprints suggested by this thesis. The numbers are random, i.e., they do not reveal the actual velocity of an existing scrum team. A bar chart is drawn in (velocity, sprint)-axis to visualize the velocity of each sprint. Also Koponen (2008) utilized bar charts for visualizing velocity over the sprints. However, he did not plot the trend or average and did not highlight the minimum or maximum value. For the case company, it is useful to plot the interpreted data since the company is still on a transition phase and seeks means to follow-up the evolution of the process. Hence, the minimum and maximum are to be highlighted to visualize the deviation. The trend is to be plotted to visualize enhances of velocity over the time and to provide insights for the estimation of the velocity for the coming sprints.

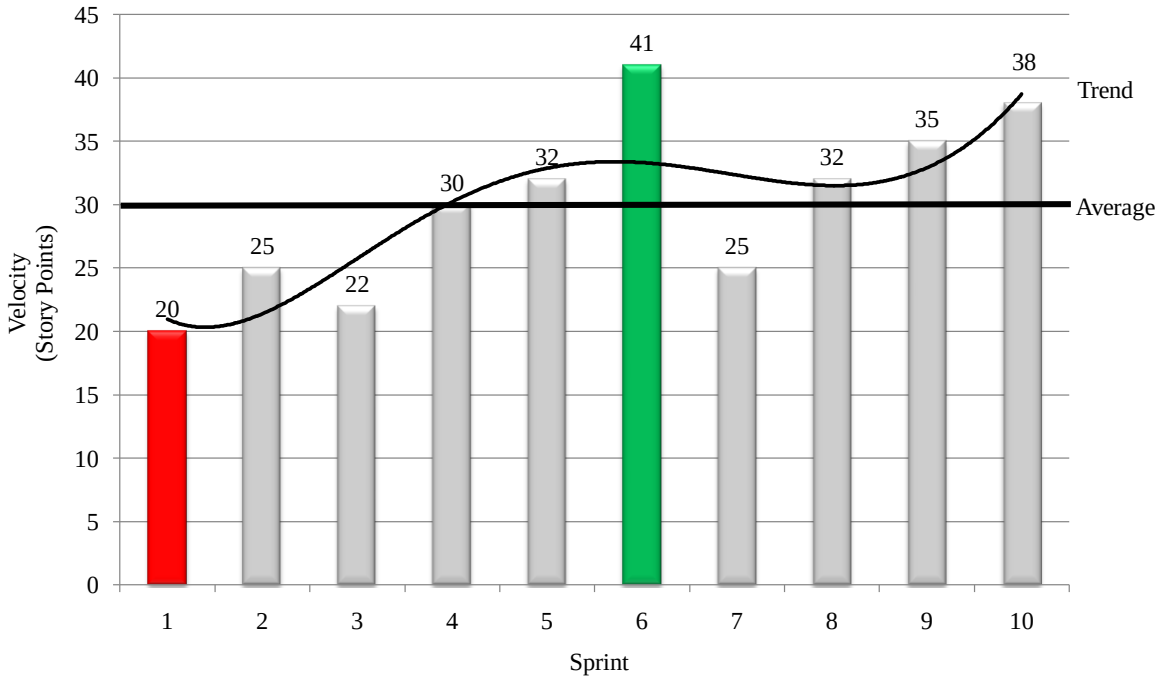

Figure 24. The suggested team level tracing for the velocity over the sprints

The unit for the velocity is the completed story points. The minimum and the maximum velocities are highlighted with red and green colors in the suggested graph. An average over all of the sprints is calculated and visualized with a line. In the example, the average is 30 story points per sprint. In addition, the current trend is visualized to support the estimation of velocity.

It is important to note that the polynomial order of the trend curve impacts on the elasticity of the curve. The curve in Figure 24 is fifth order. Since the function is fifth order, the curve reacts to the changes in the velocity on a decent level. If, for instance, a second order function would be plotted, the curve would reveal only a high-level trend since the sign of the tangent could not be changed more than once. As opposite, if too high order function is involved, the curve would be too elastic for the peaks and too complex to solve. However, solving and plotting higher than two order function is to be done with the help of software, for instance, Microsoft Excel spreadsheet. It is important to emphasize that the velocity is not to be utilized as a metric of the team. Thus, only the team should be aware of the velocity data discussed so far.

According to the interviews, other stakeholders should be aware only of the feature level velocity, i.e., the sum of the velocities of the teams developing the same feature. Since the velocity is traced in terms of tasks within a sprint, the feature level velocity can be calculated only at the end of the sprint. Once the sprint ends, the tasks are mapped to the user stories, from which the tasks were originally interpreted, updating the status of the product backlog. Thus, the cycle of the feature level velocity is two weeks.

The feature level velocity can be utilized to estimate the date of feature completion and thus to estimate the scope of a product release. The feature level involves the aspects of scaling agile to a large extent. For instance, the throughput is a function of the number of the scrum teams. However, over the study, there was no empirical data available in the case company regarding the impacts of the scaling. For instance, it was unsure that if the average velocity of the pilot teams is x story points, will the average of the velocities of all teams be equal, greater than or less than x story points post scaling.

However, defining the feature level velocity is not as straightforward as presented. Indeed, the story points of different teams cannot be summed without coefficients. Each scrum team estimates the story points for the user stories. Hence, the number of story points is not comparable between the teams. The velocities are to be multiplied with proper coefficients before summing. The procedure is similar to calculating with foreign currencies – the figures are to be stated in a common currency prior to summing up the balance.

Overall, the velocity enhances the transparency and predictability of the new feature development. In addition, it can be utilized to visualize the progress of completing the user stories of the product backlog. However, according to the interviews and to Heidenberg (2010), the velocity is not to be involved as a metric for the team performance – instead, it is to be utilized as a tool for estimating.

5.2 Work to be Done and Date of Feature Completion

The velocity and the product backlog can be involved as inputs to the progress follow-up and to estimation. The Scrum framework (Schwaber, 2009) suggests implementing a sprint burndown chart and a release burndown chart to visualize the work to be done. According to the interviews, in the context of the Mobile Media Gateway, the release

burndown chart is implemented in a form of a feature burndown chart. The theory was discussed in Chapter 2.2.4.

A sprint burndown is internal data for the scrum team. The chart is updated on the whiteboard in a daily basis. The warning signs are visible in the chart if the team has selected too many or too few items for the sprint. The sprint burndown chart and the velocity within a sprint are interconnected whereas the product backlog and the feature burndown chart are interconnected.

Part of the unplanned user stories might be mandatory for completing the feature. Thus, the completed but unplanned stories cannot be regarded as waste in all of the cases. However, the additional stories prevent the teams of performing the selected stories. Hence, technical dept increases. The technical debt is generated if the team performs less story points that the idealized line suggests (Eskelinen et al., 2010).

The technical debt (or surplus) is the difference between the idealized line and the actual line in the burndown chart. The value of the idealized line at the sprint n on the team or on the feature level can be interpreted by dividing the initial number of story points (US_0) by the number of sprints (N) and multiplying with n. Thus, the cumulative technical debt can be interpreted with equation

$$Cumulative\,debt_n = \frac{US_0}{N} \times n - \sum_{1}^{n} US_{completed,n} \,. \tag{2}$$

If the sum of the completed user stories discards the completed but unplanned stories, the equation (2) holds. However, if the completed but unplanned stories are included in the sum, also the number of initial user stories is to be adjusted, i.e., the completed but unplanned stories are added to the initial number of user stories. Hence, the equation (3) would be written in form

$$Cumulative\,debt'_n = \frac{US_0 + US_{unplanned}}{N} \times n - \sum_{1}^{n} US'_{completed,n} \,. \tag{3}$$

In the case of equation (3), the idealized line becomes dynamic. With the help of the equation (2) or (3), the technical debt could be plotted as a function of the sprints completed.

As stated, in the context of the study, the release burndown chart is replaced by a feature burndown chart. The feature burndown chart is in the interest of all the stakeholders, since it reveals whether the feature will be completed within the initial schedule. If the initial schedule is compromised due to the increased number of story points or due to lower velocity than expected, the options are to postpone the feature to the next product release, change the product release date, or to narrow the scope of the feature. If it is assumed that the first option is discarded since it increases the time-to-market of the feature more than the other options, there are two options to proceed.

First, if the product release date is changed, the graph suggested by Koponen (2008), visualized in Figure 13, is valid for plotting the feature burndown chart. The graph defines a range for the date of feature completion, i.e., it defines the estimate and upper and lower boundaries for it.

Second, if the scope of the feature is elastic, the management of the new product development can be enhanced by visualizing the technical debt of the product backlog. Table 6 illustrates a product backlog, which includes two user stories that will not be completed if the technical debt is not decreased or the product release date is not postponed. The stories not to be completed are marked with grey. In addition, a row indicating the new estimate for value is added.

Table 6. The product backlog (modified from Deemer et al., 2008) enhanced by indicating the affects of the technical debt

Priority	**Item**	**Details**	**Estimate of Value**	**Initial Estimate of Effort**	**New Estimates of Effort Remaining at end of Sprint**				
					1	**2**	**3**	**4**	**...**
1	Item 1	www.company.com/PB/1	20	40	0	0	0		
2	Item 2	www.company.com/PB/2	15	25	25	0	0		
3	Item 3	www.company.com/PB/3	18	35	35	15	0		
4	Item 4	www.company.com/PB/4	5	15	15	15	0		
5	Item 5	www.company.com/PB/5	8	10	10	10	10		
6	Item 6	www.company.com/PB/6	7	15	15	15	15		
...	...	...	...	...	...	...	...		
100	Item 100	www.company.com/PB/100	5	8	8	8	8		
101	Item 101	www.company.com/PB/101	2	8	8	8	8		
102	Item 102	www.company.com/PB/102	3	15	15	15	15		
		Total	**300**	**560**	**520**	**472**	**442**		
		To be completed	**295**						

The chapter has so far discussed the work to be done and the estimate for the date of feature completion in terms of the product backlog and the burndown charts. Next, a concept named ‘the parking lot’ is introduced to feature portfolio status follow-up. Barton et al. (2008) summarized the findings of a scrum gathering. One of the fundamental finding was a dashboard for the executives. The dashboard included a parking lot. Figure 25 visualizes the concept.

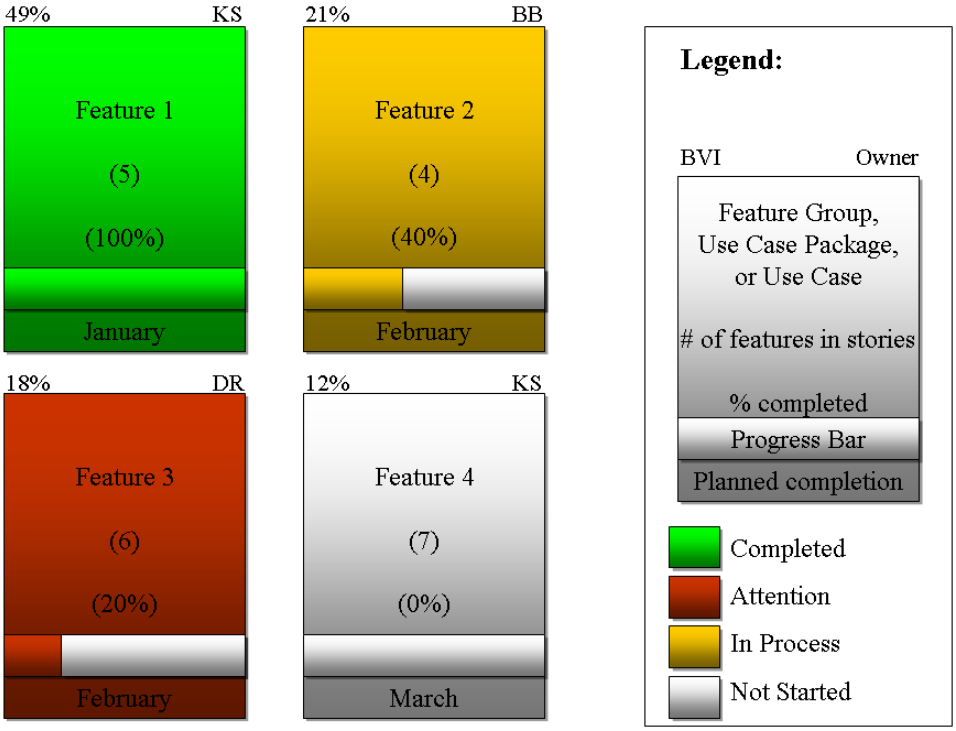

Figure 25. The parking lot concept (modified from Barton et al., 2005)

In the concept, each feature or feature group is represented by a rectangle. The color of the box depends on the status. The options for status are completed, attention, in process, and not started. The legend in Figure 25 clarifies the concept of the parking lot. Left on the top of each rectangle the Business Value Index (BVI) is stated. The index indicates the share that the feature or the feature group contributes of the total value of the product. Hence, the BVI is scaled to 100%. Right on the top, the owner of the feature group is stated. Within the rectangle, the name of the feature group, the number of the included features, and the completion percentage is indicated. In addition, the completion percentage is visualized with a progress bar. An estimate for the schedule is stated at the bottom of the rectangle. (Barton et al., 2005)

The parking lot concept intuitively illustrates the status of the release. If the concept would be implemented, the product owner team, the release management, and the product management could interpret an overview of the current status of all features with one glance. According to the interviews, one of the main interests of the release management is to receive the information regarding the appropriateness of the estimated schedule. Thus, the thesis suggests implementing the parking lot concept to address the needs of the stakeholders.

5.3 Dependencies

In the context of the Mobile Media Gateway development, dependencies emerge in multiple directions. The dependencies can be divided into three categories; product internals, company internals, and company externals. The product internal dependencies emerge for instance with other features, with other processes executed in the product,

between different scrum teams, and between hardware and software. The company internal dependencies emerge with the product portfolio, with the platform, and with the other R&D sites. The company external dependencies emerge with the acquired third party software, with the standards, and with the products of the other network equipment vendors. In the telecommunications domain, the dependencies are impossible to avoid due to, e.g., the embedded nature of the software and the interoperability requirements.

As the dependencies cannot be removed, they need to be managed. According to the interviews, the dependencies are initially identified with the help of the Early Phase Program and the feature study concept. The systemization on the product level is needed. Also Kettunen (2009) emphasized in his doctoral dissertation the need for systemization regardless of implementing the agile methods.

It can be concluded from the interviews that the stakeholders require different views to the dependencies. The scrum teams need to know the dependencies both at the user story and the task level for planning the sequence of actions. The user story level dependencies are identified in order to be able to form a feasible selection in the sprint planning meeting. In addition, the identification of the task level dependencies is needed for performing the tasks in a feasible order within the sprint and to enable the other teams to proceed in their tasks. According to the interviews, the product owner team needs to be aware of the dependencies at the user story level, for instance, to place the proper priority in the stories.

As discussed in Chapter 4.6, the release verification testers support the scrum teams in executing the early system test. Thus, the dependencies emerge between the scrum teams and the release testers. In addition, the post-GA activity experts need to be aware of the impacts of the new feature on the products that currently exist in the field.

Hence, data regarding the dependencies is needed in multiple levels. Koponen (2008) and Babinet and Ramanathan (2008) emphasize the importance of visualizing the dependencies. As a matter of fact, Babinet and Ramanthan (2008) introduced a concept of the team dependency diagram that was discussed in Chapter 2.4.3. In the diagram, the inter-team dependencies are illustrated by drawing a circle to present each team. Arrows are drawn to visualize a dependency.

The nature of the tasks is intra-team and intra-sprint. Hence, the diagram by Babinet and Ramanathan (2008) could be involved on the user story level or on the feature level. In the first option, all of the interconnected user stories in the product backlog would be drawn as circles to the diagram and linked with arrows. In the second option, the circles would present the features. If the diagram is to be involved on the task level, only the tasks within one sprint could be mapped since the tasks are not visible over the sprints.

By mapping the emerging dependencies, the bottlenecks can be identified. In addition, the software development becomes transparent and the idle time is minimized since the impediments by the dependencies can be identified and removed.

5.4 Costs and Business Value

Scrum is said to lower costs by, e.g., revealing faults early (Eskelinen et al., 2010), and to increase income by enhancing adaptability and time-to-market (Deemer et al., 2008; Schwaber, 2007; Sutherland, 2004). Hence, the case company expects to increase the profit in the long run even though the transition phase is expensive and time-consuming. According to the interviews, the investment decision of a new feature is formed at F2 by the product management. The decision is based on the estimates of, for instance, the market demand, the feasibility of the feature, the costs of development, and the expected revenues. The product management calculates the net present value based on the estimates.

The current suggestion is that the product owner team is responsible for the budget. Hence, the product management and the product owner team are the ones mainly interested in calculating the costs and the business value. There are multiple options for tracing the costs as well as for estimating the revenues. For instance, Rawsthorne (2008) suggests tracking the hours spent on implementing each user story to calculate the cost of implementation in euro per story by multiplying the hours spent with the average price per man-hour.

Regarding the revenues, Agile42 (2009) suggests the concept of return on investment factor (ROIF) as an alternative for return on investment (ROI). While calculating ROI, the investment is to be known in euro to be able to less the investment from the final value. To enhance the feasibility in the context of the Scrum framework, the return on investment factor is introduced. The difference is that the investment is to be known only in story points. The equation for ROIF (Agile42, 2009) is

$$ROIF = \frac{Business\,value}{Story\ points}. \tag{4}$$

Since the ROIF is a relative number, it can be utilized for prioritization between the features. If the ROIF of a feature is higher than of another feature, it will be prioritized on top of the other feature. If the value for ROIF is high, the feature is valuable or it is rapid to implement (Agile42, 2009).

In the case company, the product owner team and the product management could be interested in the ROIF calculations. The concept provides them a useful tool for rapid the estimation of the profitability of the new features and more over, a rapid tool for prioritizing between the features without the need for interpreting the costs or incomes in euro.

Barton et al. (2005) suggests plotting a curve of the cumulative business value including the actual earned business value in each sprint and the estimated business value. Interestingly, the estimated curve follows the pattern of the ‘S’ curve.

The concept of the continuous business value plotting is similar to the burndown chart. The cumulative business value chart illustrates both the ideal and the actual cumulative

value over the development process. By comparing the idealized and actual curve, the difference is intuitive to interpret. However, the business value of each sprint is to be calculated to plot the curve of the actual earned business value. (Barton et al., 2005)

Each feature includes an estimate for the business value and for the number of story points. Hence, the contribution of a single sprint to the feature, i.e., BV_{sprint}, can be estimated with equation

$$BV_{sprint} = \frac{Story\ points\ in\ sprint}{Story\ points\ total} \times BV_{feature} \tag{5}$$

If the sprints are synchronized, i.e., the scrum teams start and stop the sprint on the same day, the chart reveals the status of the business value of the feature in two week cycles. In addition to plotting the graph, the motivation of the scrum team is enhanced since they can interpret the business value of their activities within a sprint. However, as stated, the costs and earnings are mainly in the interests of the product management and the product owner team.

6 Conclusions

The chapter summarizes the findings of the thesis. First, the findings regarding the stakeholders and the decision framework are concluded, followed by the findings regarding the direct and indirect data that the product backlog provides. Hence, the first research question is answered in Chapter 6.1 and the second question partly in Chapter 6.1 and partly in Chapter 6.2.

6.1 The Stakeholders and the Decision Framework

In Chapter 4, each stakeholder was discussed separately. Also the decision framework was discussed. To formulate a solid understanding, it is feasible to summarize the findings to the matrix that has already been described and utilized as a framework for the semi-structured interviews. From the systems intelligence (Hämäläinen & Saarinen, 2007, pp. 3-38) perspective, it is crucial to understand the whole and interpret it from multiple angles instead of focusing on parts as has been done previously in this thesis. Prior to proceeding, it is important to emphasize that the findings are based on the interviews over the pilot phase of the implementation of Scrum. Hence, the interviews did not reveal the ultimate solution. Indeed, the interviews revealed the current thinking regarding the solution.

Nerur (2005) emphasize that the agile software development methods are guided by the features instead of being guided by the tasks and activities. Similarly, the new decision framework, and thus the product development, is guided by developing new feasible features. Kettunen (2009) divide the actions taken in another major network equipment vendor into pre-game, development, and post-game activities. Interestingly, the decision points in the decision framework can be mapped to the framework that Kettunen presents. The development is completed between decision points F2 and F4. Hence, decision points from F0 to F2 can be stated to be pre-game whereas the release management and the post-GA activities can be stated to be post-game.

According to Karlström and Runeson (2005), the Scrum framework and the state-gate model can be combined. However, Larman and Vodde (2004, pp. 208-210) discourage such a combination as discussed in Chapter 2.3. Clearly, the discussed decision framework is similar to the Cooper state-gate model (Cooper, 2000). Table 7 illustrates the similarity. Both the state-gate model and the decision framework begin by discovery and early investigations. The further the development proceeds, the more accurate investigations are needed. Interestingly, according to Table 7, the decision to develop is formed at the same point. However, the state-gate model separates the development and testing whereas the Scrum framework combines these activities. As discussed in Chapter 2.2, one of the fundamentals of Scrum is to complete the coding and the testing simultaneously. It is important to note that the study examined only the procedure in which the development of the feature is continued at each decision point.

Table 7. Comparison of the State-Gate model (Cooper, 2000) and the decision framework

The State-Gate Model (Cooper, 2000)		The Decision Framework	
	Discovery stage		Early investigation by EPP
Gate 1	Idea screening	F0	Request for one-pager
State 1	Preliminary investigation		Conduct a one-pager
Gate 2	Second screening	F1	Place the item to the product backlog
State 2	Detailed investigation		Feature Concept Study (FCS)
Gate 3	Decision to develop	F2	Decision to implement
State 3	Development		Implement with Scrum
Gate 4	Decision to test	F3	Commitment
State 4	Test		Implement with Scrum
		F4	Done
	and validate		Release project and release verification
Gate 5	Decision to launch		General Availability
State 5	Launch		
			Post-GA activities

In Table 7, the state four is divided into two rows since in the decision framework, the function testing and the early system testing is executed between F2 and F4. However, the release verification, similar to validation, is executed in the release verification phase. The fifth gate and the fifth state are both bundled to the point of General Availability.

While discussing the investigations, it is to be noted that literature (Kähkönen, 2004; Sutherland, 2008) is solid in stating that only lightweight documents are to be written. Although, Kettunen (2009) recognized that at least in the context of the network equipment vendors, the architecture is to be defined. Based on these, it can be concluded that the one-pagers are aligned with the literature. However, it can be argued whether the feature concept studies are aligned or not. The framework suggests that the FCS is to be as light-weight as possible. As opposite, the interviewees discussed that the study includes also detailed description.

Racheva and Daneva (2008) encourage utilizing the real options analysis on top of the agile software development models. The decision framework can be examined also through the lenses of the real options analysis. First, only tens of hours are spent on writing a first description of the feature, i.e., the one-pager. Next, preliminary investigations are executed. If the outcome of the investigations is positive, the feature is placed in the product backlog. The feature concept study is performed by the product owner team before finalizing the investment decision. Hence, the development proceeds stepwise providing an option for terminating the process after relatively small investments are made.

Uncertainty decreases as a function of time in new product development as stated in the concept ‘Cone of Uncertainty’. The vertical axis can reveal for instance the business value or the estimate for effort. The initial estimate varies from 0.25 to 4. After each phase, the uncertainty is narrowed. Once the software is released, the business value or effort needed is fixed, i.e., the cone narrows to one. (Boehm, 2008) Figure 26 illustrates the cone.

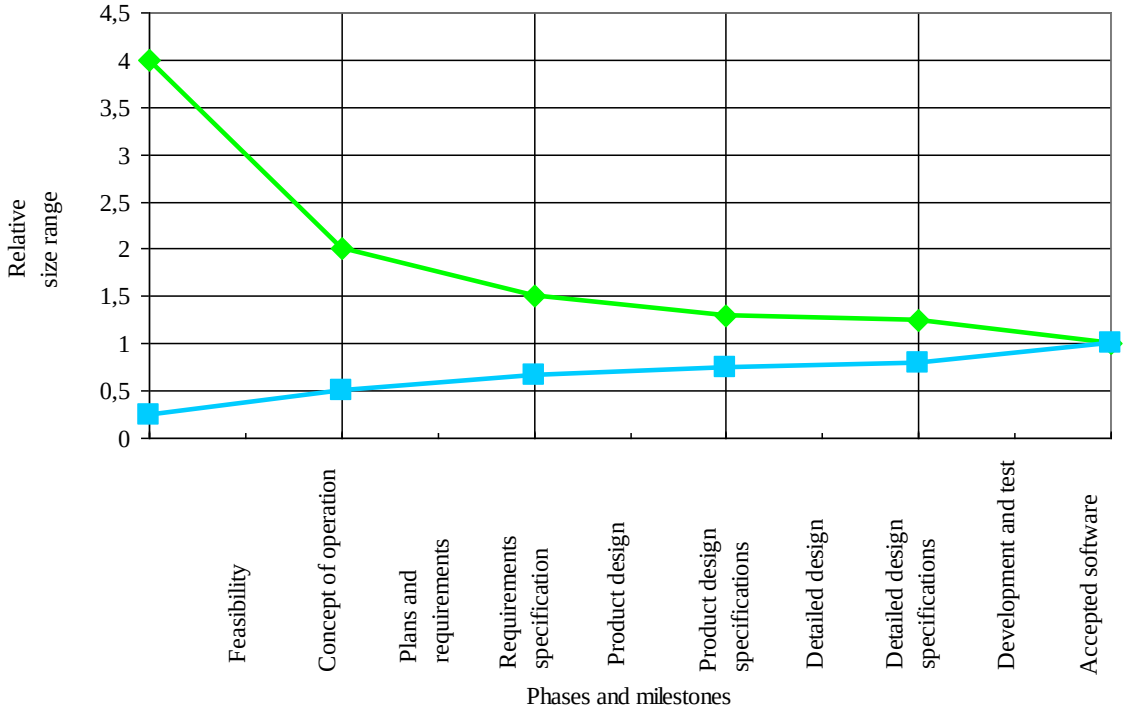

Figure 26. Cone of Uncertainty (modified from Boehm, 2008)

The cone can be applied to the decision framework. The phases and milestones in the Cone of Uncertainty can be changed to the decision points from F0 to F4. Hence, there is a clear similarity between the decision framework and the cone.

Tengshe and Noble (2007) emphasize the importance of an approval matrix that identifies the authorized persons. The approach in the study was to form a matrix identifying the stakeholders and their activities at each decision point. According to the research questions, a special focus was on the activities regarding the product backlog. To add value for the case company, reflections from literature were added to the matrix. The supporting findings from literature were stated as ‘positive’ and the opposite findings were stated as ‘negative’.

From literature (Chapter 2), the stakeholders were identified to be the scrum teams, the scrum masters, and the product owner. Also, multiple authors discussed the role of management. However, the management structure varied among the companies. Additionally, stakeholders were identified from the open-ended interviews. The identified stakeholders were as follows: the innovation process, the Early Phase

Program, release verification testers, release management, and the post-GA activities. Besides identifying new stakeholders, the interviews revealed that the case company involves a product owner team instead of only a single product owner.

Interestingly, according to the interviews, the activities between F2 and F3 are similar to activities between F3 and F4 for multiple stakeholders of the product backlog. The scrum teams, the product owner team and the scrum masters perform similar actions in both phases. Additionally, the support provided by the release level testers and the innovation process are currently similar in both phases.

The findings from the interviews were marked to the matrix discussed previously in the thesis. In addition, the findings from the literature review were stated. Appendix D presents the formed matrix. The findings have been discussed previously in Chapter 4 stakeholder by stakeholder. However, some relevant high-level conclusions can be interpreted by focusing on the overall matrix instead of only one stakeholder. Next, three main positive findings are discussed.

First, the customer requests and the new innovations flow through the Early Phase Program, which is aligned with the opportunity team concept introduced by Nahor and Katzav (2009). The decision framework recognizes the likeliness of changes in the customer requests. The agile manifesto (Fowler & Highsmith, 2001) is based on the fact that the customers change their behavior and interests. At Nokia Networks, adaptability to the changes in the customer requests increased by 69% due to the agile software development methods. However, it the Early Phase Program is isolated from the scrum teams, knowledge gaps may emerge along with long feedback loops. Hence, the cooperation between the developers and the EPP team is to be enhanced.

Second, the concept of the minimum marketable feature is to be mentioned. The concept suggests that a portion of the new feature is to be discarded from the first release to enhance the time-to-market and the throughput of the scrum teams. The initial release includes only the most important functionalities of the feature. The MMF concept is aligned with lean; Raman (1998) states that the new features are to be pull by the customer whereas Benefield (2008) and Poppendieck (2002) states that there should be nothing extra in the new features to avoid waste.

Third, the product owner responsibility is shared. As identified by Larman and Vodde (2009) as well as by Lowery and Evans (2007), only one product owner will become too busy. Hence, the requirement of maintaining the product backlog prioritized all the time is compromised.

Next, the three most notable findings regarding areas to be enhanced are discussed. The first finding is that currently the innovation process is separated from the new feature development. The current thinking is that there is to be a separated innovation backlog. Once Scrum is scaled to the large extent, the new ideas are inserted into the system from the innovation process. Schilling (2004, pp. 4-5) discusses an innovation funnel, i.e., a filter that selects and combines the most feasible ideas and converts them to

innovations. Currently, the funnel is only an input system to the new feature development.
This thesis suggests integrating the innovation funnel (Schilling, 2004, pp. 4-5) and the decision framework. Figure 27 illustrates the funnel as input to the decision framework. It also illustrates the suggested integration.

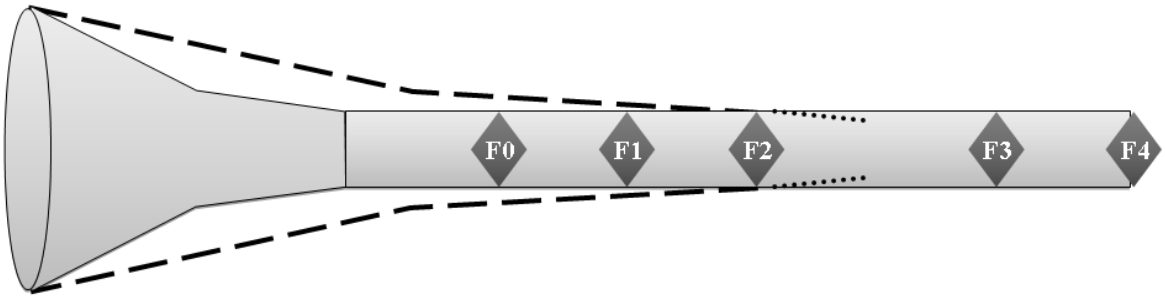

Figure 27. The innovation funnel (Schilling, 2004, pp. 4-5) combined with the decision framework

The investment decision is formed at F2. Thus, in the future, the contribution of the innovation process could be extended, i.e., the decision framework and the innovation process could be merged. The dashed line in Figure 27 visualizes the potential extension of the innovation process. With the extension, the innovation funnel would cover the whole process of converting ideas to innovation.

It can be argued whether the funnel could even continue until between F2 and F3 (illustrated with the light dashed line in Figure 27). In this case, the scrum teams would, e.g., code pilots of a feature. Closer to F3, one of the pilots would be selected as the solution while the other pilots would be discarded. The discarded pilots cannot automatically be categorized as waste since they can provide valuable insights into the ultimate solution.

The second finding, regarding areas to be enhanced, is the current thinking of utilizing multiple backlogs. According to the interviews, the desired situation by many stakeholders is to involve their own backlog. In total, according to the interviews, at least five backlogs are to be implemented; a product backlog for the scrum teams, a backlog for the trouble reports at the post-GA phase, a backlog for the customer requests in the post-GA activities, a backlog (or list) for one-pagers and an innovation backlog. In addition, the release verification testers might implement a separated backlog for their test cases.

Literature (Laanti, 2008; Hornos & Izquiero, 2009; Racheva & Daneva, 2008; Sutherland, 2008) clearly states that involving only one product backlog is encouraged; yet, the backlog needs to utilize multiple levels to meet the needs of all stakeholders. However, de la Rie (2008) identified that the support, i.e., the post-GA activities in the context of the case company, may involve a separated backlog and a separated support team to protect the scrum teams, which are developing new features, from the external interfaces. According to Racheva and Daneva (2008), also the ideas are to be placed in

the product backlog. Hence, the Scrum framework includes a built-in innovation process since the product backlog items can be split, reprioritized and deleted based on the emerging knowledge (Barton, 2009). The innovation process can be involved only by combining the frameworks as suggested in Figure 27.

The findings suggest combining the backlogs. The backlogs for the post-GA trouble reports and customer requests could be merged. However, the thesis has not studied the nature of these two backlogs comprehensively since the focus was on the new feature development.

The innovation backlog and the one-pager list are to be merged with the product backlog. With this solution, the emerging ideas are placed in the product backlog and prioritized along with the items for completing the features. Initially, the ideas are uncertain in a sense of business value and costs. Additionally, the descriptions and definitions are unspecified. Hence, the new ideas are large items with low priority. The product owner team should create smaller items for evaluating the ideas. Figure 28 illustrates the concept of merging the innovation backlog and the product backlog.

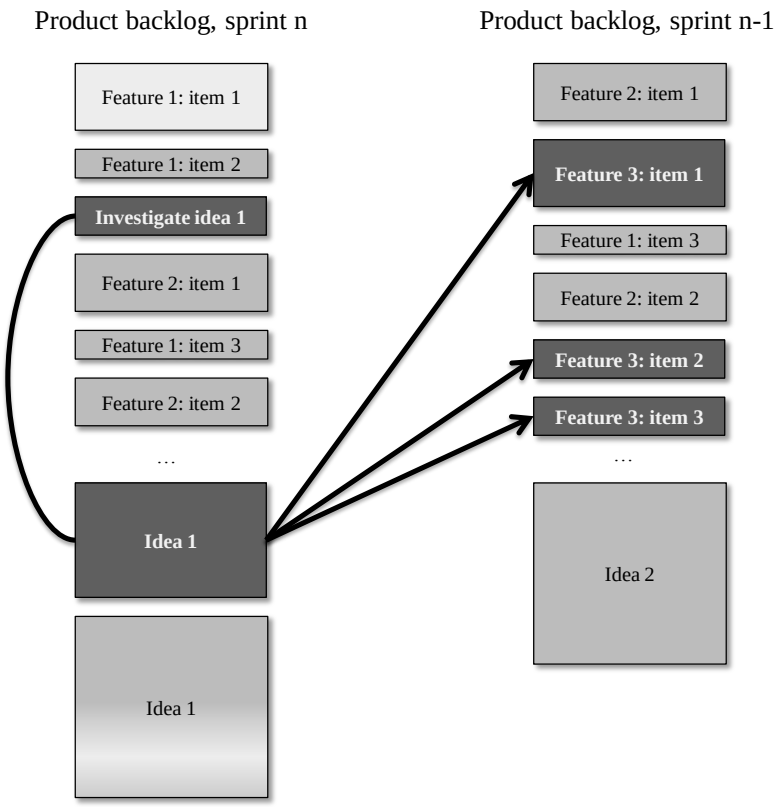

Figure 28. The innovation backlog bundled to the product backlog

The third finding regarding areas to be enhanced is related to the handovers. According to Benefield (2008) and Poppendieck (2002), the handovers create waste and thus are not aligned with the lean principles. In total, five handovers are conducted in the framework. First, the customer requests from the product management, or the new ideas

from the innovation process, are utilized as input to the EPP team. Second, the EPP team hands over the new feature to the product owner team. Third, the scrum teams implement the feature based on the feature concept study by the product owner team. Fourth, at F4, the new feature is transferred from the scrum teams to the release management, to be more precise, to the release verification phase. Fifth, once the feature is released to the markets, the post-GA support experts receive the responsibility over the feature.

The handovers also generate knowledge gaps as identified by Kettunen (2009) and Woi Hin (2006) since the people receiving the responsibility receive only the summary of the previous discussion and research. Especially, a severe gap may emerge if the scrum teams are not familiar with the requirements (Woi Hin, 2006). However, there are remedies for the gaps. For instance, Auvinen et al. (2008) encourage the scrum teams to split the product backlog items. Additionally, Schwaber (2009) supports the product backlog grooming sessions held by the scrum teams. Overall, the knowledge sharing is to be enhanced and the silos of knowledge are to be avoided (Kalliney, 2006).

According to the interviews, the scrum teams attend the product backlog grooming sessions every other week. Thus, the product backlog is updated continuously. Hence, the product backlog is dynamic as suggested by Schwaber (2009). The product backlog is to be ready all the time to enhance the process of selecting items (Deemer et al., 2008).

The number of handovers can be decreased by involving different stakeholders early in the process. For instance, by involving the proxy product owners at least partly in the Early Phase Program, the second handover can be smoothened. In a similar manner, involving the scrum teams in the feature concept study softens the third handover.

Overall, it can be stated that the decision framework includes both key success factors and issues to be enhanced. The chapter discussed the high level findings of the framework followed by conclusions regarding the three most notable positive and negative findings.

6.2 Product Backlog Direct and Indirect Data

Chapter 5 discussed the data the different stakeholders need and the means of visualizing the data directly and indirectly provided by the product backlog. The chapter relied on interviews and literature. However, also new models were suggested. Table 8 summarizes the suggested means of visualization. The table is a suggestion for a data visualization framework in the development of the Mobile Media Gateway.

The velocity is to be calculated on the feature level and on the team level. The team level velocity is internal data for the team. It can be divided into the velocity calculated within the sprint and over the sprints. The visualization within the sprint is conducted with the help of the sprint burndown chart visualized in Figure 9 (Deemer et al., 2008). Figure 24 illustrates the suggested model for visualizing the team velocity over the

sprints. The feature level velocity is the sum of all team level velocities. The data can be visualized with a feature burndown chart, similar to the sprint burndown chart.

The flow of gaining the velocity data could be as follows. The scrum teams estimate the number of story points for each item in the product backlog grooming sessions prior to the sprint. At the end of the sprint, the sum of the user stories related to the completed items is calculated by the team and the scrum master. This data is utilized in the team retrospectives. The scrum masters sum up the team level velocities in the Scrum-of-Scrums meetings. The sum is the feature level velocity. The scrum masters present this data to the product owner team, which in turn, presents the data to the product management in addition to the budget follow-up and other figures.

The technical debt is as well measured in two perspectives. Again, the team level data is internal to the team whereas the feature level debt is in the interests of all stakeholders, especially of the product owner team. The debt is visible in the burndown charts but can also be calculated with equations (2) and (3). The results of the calculations can be plotted to form a figure of the technical debt only, i.e., without the burndown chart. In addition, if the scope is elastic, the thesis suggests marking the debt to the product backlog according to Table 6.

The feature status can be visualized with the feature burndown chart whereas the overall release status can be visualized with the parking lot concept introduced by Barton et al. (2005). Both of these statuses are in the interests of all stakeholders, especially of the product owner team and the release management. If the schedule is elastic, the date range concept by Koponen (2008), illustrated in Figure 13, is feasible.

The management of the dependencies was a concern of multiple interviewees. Thus, a key success factor of the scaling of Scrum is to properly manage the emerging dependencies that might not be self-evident. One option is to discuss the dependencies in the Scrum-of-Scrums meetings (Lowery & Evans, 2007; Babinet & Ramanathan, 2008). Babinet and Ramanathan (2008) introduce a framework of drawing lines between the teams to indicate the dependencies and to identify the bottlenecks. The framework could be beneficial also for the case company.

The dependencies are to be mapped within the team, i.e., on the task level, and on the user story level. For the user story level, the dependency diagram by Babinet and Raminathan (2008) can be applied by changing the mapping of dependencies between the teams to the mapping of dependencies between the user stories. On the task level, the dependencies are to be drawn in the whiteboard since the scrum teams are not utilizing a software tool for managing the tasks.

The business value of the feature can be tracked by plotting the earned value at each of the sprints on the feature level (Barton et al., 2008). By a minor modification to the concept of the continuous business value follow-up, i.e., by calculating the business value per sprint for the internal discussions in the team, the scrum team can be motivated.

Table 8. Summary of means for visualizing data

Data	Stakeholder	Visualization	Utility	Cycle
Velocity within the sprint	The scrum team and the scrum master	Sprint burndown chart (Figure 9, Deemer et al., 2008)	Know current progress	24h
Velocity over the sprints	The scrum team and the scrum master	Diagram of velocities over sprints (Figure 24)	Trend, average, min and max as input to planning and retrospective	two weeks
Velocity on the feature level	All stakeholders	Feature burndown chart (Figure 9, Deemer et al., 2008)	Estimate feature release date and scope	two weeks
Technical debt within the sprint	The scrum team and the scrum master	Equations (2) and (3) can be plotted	Understand the focus and feasibility of schedule	24h or two weeks
Technical debt on feature level	All stakeholders, especially the product owner team	Equations (2) and (3) can be plotted	Understand the focus and feasibility of schedule	two weeks
		Mark the user stories that will not be finished with grey in the product backlog (Table 6) if the scope is elastic	Estimate for the new scope	two weeks
Dependencies on task level	The scrum team and the scrum master	Draw the dependencies on the whiteboard, Koponen (2008) suggest visualizing the dependencies on the task level	Identify the proper sequence of completing the tasks	24h
Dependencies on user story level	The scrum team, the scrum master, and the product owner team	Draw a dependency map (Babinet & Ramanathan, 2008)	Identify bottlenecks and reprioritize	two weeks
Feature status over the sprints	All stakeholders, especially the product owner team	Feature burndown chart (Deemer et al., 2008)	Know current progress	two weeks
Overall status of the release	All stakeholders, especially the release management and the product owner team	Parking lot (Figure 25, Barton et al., 2005)	Provide the status of the release	two weeks
The range for the date of feature completion if schedule is elastic	All stakeholders, especially the release management and the product owner team	Plot boundaries for the estimate for the date of feature completion (Figure 13, Koponen, 2008)	Know the limit within the date may change	two weeks
Earned business value per sprint	The scrum team and the scrum master	Plot earned business value per sprint (Equation 7, Barton et al., 2008)	Know current progress and enhance the motivation of the team	two weeks
Earned business value per feature	All stakeholders	Plot earned business value per feature (Barton et al., 2008)	Know current progress	two weeks
ROIF	The product owner team and product management	Result of Equation (4), (Agile42)	Estimate the revenue and prioritize	two weeks

7 Discussion

The thesis was written at Ericsson Finland R&D in the context of development of the Mobile Media Gateway. The organization is planning the transition from the current pilot scrum teams to the large-scale implementation of Scrum. Hence, it is to be noted that this thesis discusses the current thinking at the organization, not the ultimate solution.

The objective of the research was to identifying the stakeholders and their activities in addition to the data they utilize at each point. The focus was on the product backlog. The mapping of the stakeholders and activities was provided in a form of matrix (Appendix D), illustrating all the decision points of the decision framework. The matrix also discussed the needed data at different decision points. However, the discussion regarding the needed data was concluded by providing a framework in Chapter 5 for visualizing the data for different stakeholders.

The contribution of the thesis can be divided into two categories; contribution to the case company and to the academic society. To the case company, the thesis provides an analysis of the actions of each stakeholder at each decision point from the product backlog perspective. In addition, the thesis briefly suggests the means of visualizing data. Both of these topics are important to design the proper implementation of the product backlog.

The organization has gained more knowledge from the pilot teams than it had at the beginning of the study. However, the research questions are still valid, i.e., solutions to the questions were not fully known prior to completing the thesis. An additional outcome of the thesis was a high-level description of the decision framework, which has previously been described only in slide sets, and the conducted benchmark.

To the academic society, the thesis provides a case example of implementing Scrum in a large-scale organization. Especially, the combination of the state-gate model and the Scrum framework is discussed from the product backlog perspective. Only a limited number of publications discussing the combination were discovered in the literature review. In addition, the thesis provides an example of approaching study through a decision framework.

The study relies on the open-ended interviews (n = 6), on the semi-structured interviews (n = 11) and on the literature review. The findings from the interviews represent the current thinking of the stakeholders since there was no ultimate solution. Hence, the involved qualitative data was prone to bias. The reliability of the study could have been improved by increasing the sample size. However, the chosen interviewees were key people in the pilot and transition phase. The interviewees could provide valuable insights that the other people in the organization would not. It is also important to note that interviews decrease the time the employees of the case company can spend on their work. The larger the sample size, the more employees are interrupted.

The literature review was based on academic papers, publications by the Scrum Alliance, and publications of other telecommunication companies. The Majority of the involved publications were written by recognized and experienced professionals in the field of software development models. The sample size for the literature review was relatively high. However, the debate regarding the Scrum framework in the articles, books, and internet has enormous magnitude. Hence, the thesis provides only one perspective for the current literature.

The research was tailored to the case company. Hence, the generalizability of the results regarding the identification of the stakeholder activities at each decision point is low. However, the results of the data each stakeholder need can be applied also to other companies since at least other large companies likely involve similar stakeholders in the organization. In addition, the method of the study can be utilized also in other contexts.

Based on the thesis, a need for further research in the academic society as well as in the case company can be identified. From the academic perspectives, at least two aspects are to be studied. First, according to the literature review, the combination of the Scrum framework and the state-gate type of approach is both positive (Karlström & Runeson, 2005) and negative (Larman and Vodde, 2004, pp. 208-210). Hence, further research on the affects of the combination is needed. Second, according to the literature review, the means of enhancing the innovativeness in the Scrum framework is not widely discussed.

In the context of the Mobile Media Gateway development, three issues are to be examined and, if possible, enhanced. First, according to the current thinking, multiple backlogs are to be implemented. The approach violates the findings of the literature review. Hence, further discussion is needed on the means of merging the backlogs, e.g., with the help of a more hierarchical product backlog. Second, the decision framework includes multiple handovers that are not aligned with lean. Hence, study is needed to remove the handovers to enhance the lean thinking. Third, the innovation process is currently treated separated from the Scrum implementation. Obviously, a key success factor in the wide perspective is the innovativeness. Hence, the development of new features with the Scrum framework is to be merged with the innovation process.

The scaling phase will continue in the case company. The conclusions from the thesis will be distributed to the key people involved in the scaling. To maximize the added value of the thesis, it has been agreed that the matrix of stakeholder activities will be review during December 2010 to interpret whether the identified negative aspects have been minimized and the positive aspects maximized.

Overall, it can be stated that the study provides a solution for the research questions. The stakeholders of the product backlog were identified and their activities were mapped to the decision framework. Additionally, the data needed from and provided to the product backlog was discussed. The thesis provides a basis for enhancing the Scrum implementation of the case company to meet the goals in flexibility and efficiency the organization has stated.

References

Abrahamsson Pekka et al., 2003, New Directions on Agile Methods: A Comparative Analysis, Proceedings of the 25th International Conference on Software Engineering (ICSE'03), p. 244-254, 3-10.5.2003, IEEE ISSN 0270-5257/03, DOI 10.1109/ICSE.2003.1201204

The conference paper presents the evolution of the agile software development methods.

Advanced Development Methods Inc., 1996, Controlled Chaos: Living on the Edge, available at http://www.controlchaos.com/download/Living%20on%20the%20Edge.pdf [referred 5.1.2010]

The paper describes the origins of Scrum and the characteristics of the first Scrum framework.

Agile42, 2009, Return on Investment Factor (ROIF), web page, updated 25.9.2009, available at http://agilo.bwinlabs.net/win.com/agilo-help/scrum/roif [referred 2.5.2010]

The online guide describes the concept of ROIF.

Artto Karlos, 2010, Projektien suunnittelu: laajuus, aikataulu ja resurssit, lecture handouts of course TU-22.1120 Projektien suunnittelu ja ohjaus, 5.2.2010, Aalto University School of Technology and Science

(In Finnish) The lecture handouts discuss the project timing, time planning, and S-curve.

Auvinen Jussi et al., 2005, Improving the Engineering Process Area at Ericsson with Agile Practices, Turku Centre for Computer Science - TUCS, TUCS Technical Report No 716, October 2005, available at http://crest.abo.fi/publications/public/2005/TR716.pdf [referred 10.2.2010]

The report describes experiments of XP and Pair Programming at Ericsson in Turku.

Babinet Eric & Ramathan Rajani, 2008, Dependency Management in a Large Agile Environment, Agile 2008 Conference, 5-8.1.2009, p. 401-406, IEEE ISSN 978-0-7695-3321-6, DOI 10.1109/Agile.2008.58

The article introduces a method of visualizing the dependencies in the Scrum framework.

Barton Brent, 2009, All-Out Organizational Scrum as an Innovation Value Chain, Proceedings of the 42nd Hawaii International Conference on System Sciences, p. 1-6, IEEE ISSN 1530-1605, DOI 10.1109/HICSS.2009.57

The article discusses linking the Scrum framework and the innovation process.

Barton Brent et al., 2005, Reporting Scrum Project Progress to Executive Management through Metrics, The Agile Times, a publication of the AgileAlliance, Vol. VII, 2005, available at http://www.agilealliance.org/system/agile_times/file/1671/agile_times_7.pdf [referred 7.4.2010]

The article discusses combined findings of a Scrum Gathering in 2005. The co-authors are Dan Rawsthorne and Ken Schwaber.

Beck Kent, 1999, Embracing Change with Extreme Programming, Computer, Vol. 32, Issue 10, October 1999, p. 70-77, IEEE ISSN 0018-9162, DOI 10.1109/2.796139

The article describes the basics of Extreme Programming.

Benefield Gabrielle, 2008, Rolling out Agile in a Large Enterprise, Proceedings of the 41st Hawaii International Conference on System Sciences, IEEE ISSN 1530-1605, DOI 10.1109/HICSS.2008.382

The conference paper describes the agile implementation experiences from Yahoo! Corporation.

Boehm Barry W., 1986, A Spiral Model of Software Development and Enhancement, ACM SIGSOFT Software Engineering Notes, Vol. 11 Issue 4, August 1986, p.14-24, ACM ISSN 0163-5948

The first paper introducing the spiral model.

Boehm Barry W., 2008, Making a Difference in the Software Century, excerpt from Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research, R.W.Shelby, 2007, IEEE CS Press-John wiley & Sons, excerpt published in IEEE Computer, Vol. 41, Issue 3, DOI 10.1109/MC.2008.91

The paper describes the uncertainty of software development and emphasizes the inflexibility of the Waterfall model.

Brown Mathew et al., 2007, Rugby for Dummies 2nd Edition, John Wiley & Sons Canada Ltd.

The book is an introduction to rugby (the sport) for new players.

Cohn Mike, 2007, Guide to Transitioning, Presentation at London Scrum Gathering 2007, available at http://www.scrumalliance.org/resources/285 [referred 8.1.2010]

The presentation introduces practical guidelines for implementing and scaling agile.

Cohn Mike, 2008a, Prioritizing Product Backlog, Presentation at Chicago Scrum Gathering April 2008, available at http://www.scrumalliance.org/resources/338 [referred 8.1.2010]

The presentation describes techniques for prioritizing user stories in the backlog.

Cohn Mike, 2008b, User Stories for Your Product Backlog, Presentation at Chicago Scrum Gathering April 2008, 14.4.2008, available at http://www.scrumalliance.org/resources/337 [referred 8.1.2010]

The presentation provides insights to inserting user stories to the product backlog.

Cooper Robert G., 2000, Winning with New Products, Ivey Business Journal, Jul/Aug2000, Vol. 64, Issue 6, p. 54-61, ISSN 1481-8248

The article presents the Cooper state-gate model.

Cooper Robert G., 2001, Winning with New Products, State-Gate Inc., Ontario, Canada, revised version of article from Ivey Business Journal in 2000, available at http://aitel.hist.no/fag/mop/lek02/Winning%20with%20new%20products.pdf [referred 25.2.2010]

The revised article presents the Cooper state-gate model with new visualizations.

de la Rie Frans, 2008, Ericsson: Agile CM, Online Presentation, available at http://www.topic.nl/nl/cm-workshop/presentations/Frans de la Rie.pps [referred 9.2.2010]

The slide set presents the agile solution of Ericsson at the Netherlands.

Deemer et al., 2008, The Scrum Primer Version 1.1, Scrum Training Institute, available at http://www.scrumprimer.com [referred 5.1.2010]

This guide describes the very basics of the Scrum framework, a must to read for a novice of Scrum.

Dogson Mark et al., 2008, The Management of Technological Innovation, Oxford University Press, ISBN 978-0-19-920852-4

The book provides insights to the strategic innovation management.

Drobka Jerry et al., 2004, Piloting XP on Four Mission-Critical Projects, Software IEEE, Vol. 21, Issue 6, p. 70-75, IEEE ISSN 0740-7459, DOI 10.1109/MS.2004.47

The paper presents findings of changes in the productivity, the quality, and the test coverage at Motorola because of implementing Extreme Programming.

Ericsson, 2009, The future role of telecom, Ericsson Whitepaper 284 23-3232 Uen Rev A, September 2009, available at http://www.ericsson.com/res/thecompany/docs/whitepapers/The_future_role_of_telecom.pdf [referred 22.2.2010]

The whitepaper describes the current trends in the telecommunications domain.

Eskelinen Arto et al., 2010, Scrum Master Course, lecture notes by Reaktor Innovations Oy, Course at Ericsson, Kirkkonummi Finland 12.-13.1.2010

The lecture notes present the basics of Scrum, especially from the scrum master perspective.

Evans Ian, 2006, Agile Delivery at British Telecom, Methods & Tools Summer 2006, Vol. 14, No.2, pp. 20-27, ISSN 1661-402X, available at http://www.methodsandtools.com/PDF/mt200602.pdf [referred 10.2.2010]

The article discusses the changes at British Telecom that agile means of working introduced.

Fowler • Martin & Highsmith Jim, 2001, The Agile Manifesto, Software Development Magazine August 2001, available at http://andrey.hristov.com/fht-stuttgart/The_Agile_Manifesto_SDMagazine.pdf [referred 15.1.2010]

The paper describes the Agile Manifesto.

Goos Jasper & Melisse Alfo, 2008, An Ericsson Example of Enterprise Class Agility, Agile 2008 Conference, p. 154-159, IEEE ISSN 978-0-7695-3321-6, DOI 10.1109/Agile.2008.24

The paper describes the experiments of using the agile methods at Ericsson in the Netherlands.

Guimarães Luciano Rodrigues and Vilela Plínio Roberto Souza, 2005, comparing Software Development Models Using CDM, Proceedings of the 6th conference on Information technology education, October 20.–22.10.2005, p. 339-347, Newark, New Jersey, USA, ACM ISBN 1-59593-252-6

The article discusses the differences of software development models. The focus is in the Waterfall model and in the spiral model.

Gul Ensar et al., 2008, Using XP in Telecommunication Software Development, The Third International Conference on Software Engineering Advances, p. 258-263, IEEE ISSN 978-0-7695-3372-8, DOI 10.1109/ICSEA.2008.11

The article presents experiences of Extreme Programming in the telecommunications domain. For instance Vodafone and Nortel are involved in the case studies.

Heidenberg Jeanette, 2010, Agile Metrics, a presentation at Cloud WP2 programme, 6.4.2010, Software Business and Engineering Institute SoberIT, Innopoli 2, Espoo

The presentation discussed the metrics in the agile software development.

Holmström Olsson Helena, 2009, Acting Agile in 'Streamline Development', In Proceedings of IRIS 32 (Information Research in Scandinavia), August 9-12, Molde, Norway, available at http://nordhaug.himolde.no/myreview/files/group-7/A15.pdf [referred 10.2.2010]

The paper identifies prerequisites for implementing streamline development and discuses the enhancements the framework offered. The paper relies on a case study at Ericsson.

Hornos Gemma & Izquierdo Mónica, 2009, Agile at Telefonica R&D, Presentation at Germany Scrum Gathering October 2009, available at http://www.scrumalliance.org/resources/1105 [referred 8.1.2010]

The presentation discusses the implementation of agile in Telefonica.

Hulkko Hanna & Abrahamsson Pekka, 2005, A Multiple Case Study on the Impact of Pair Programming on Product Quality, Proceedings of the 27th international

conference on Software engineering, 15–21.5.2005, St. Louis, Missouri, USA, p. 395-504, ACM ISBN 1-58113-963-2

The paper describes findings from using Pair Programming in multiple companies.

Hämäläinen Raimo P. & Saarinen Esa, 2007, Systems Intelligence in Leadership and Everyday Life, Helsinki University of Technology, ISBN 978-951-22-88-37-3

The book is combined from an article by Saarinen and Hämäläinen and assignment papers by their students.

ITEA, 2006, Agile Software Development of Embedded Systems, Newsletter #2/2006, available at http://www.agile-itea.org/public/deliverables/D.6.4.4_ITEA-AGILE-Newsletter2-2006.pdf [referred 5.1.2010]

The newsletter discusses the implementation of agile in for instance at Nokia and F-Secure. Bas Vodde was the driver at Nokia.

Ivček Mario & Galinac Tihana, 2009, Adapting Agile Practices in Globally Distributed Large Scale Software Development, Ericsson Whitepaper, available at http://www.ericsson.com/hr/etk/dogadjanja/mipro_2009/21_1337_F.pdf [referred 9.2.2010]

The article presents findings of agile implementation at Ericsson GSD in Croatia.

Kalliney Marie, 2009, Transitioning from Agile Development to Enterprise Product Management Agility, Agile Conference 2009, 24-28.8.2009, p.209-213, IEEE ISBN 978-0-7695-3768-9/09, DOI 10.1109/AGILE.2009.64

The conference paper describes lessons of scaling agile to the enterprise level.

Karlström Daniel & Runeson Per, 2005, Combining Agile Methods with Stage-Gate Project Management, IEEE Software, Vol. 22, Issue 3, May-June 2005, p. 43-49, DOI 10.1109/MS.2005.59

The article presents findings from ABB, Ericsson, and Vodafone regarding combining an agile software development method with the state-gate project management.

Kettunen Petri, 2009, Agile Software Development in Large-scale New Product Development Organization: Team-level Perspective, Doctoral Dissertation 186, Helsinki University of Technology, available at http://lib.tkk.fi/Diss/2009/isbn9789522481146/isbn9789522481146.pdf [referred 5.1.2010]

The new doctoral dissertation from Helsinki University of Technology introduces findings form using agile in the telecommunications domain.

Kniber Henrik, 2007, Scrum and XP from the Trenches - How we do Scrum, C4 Media Inc., InfoQ Enterprise Software Development Series, ISBN 978-1-4303-2264-1, available at http://www.crisp.se/henrik.kniberg/ScrumAndXpFromTheTrenches.pdf [referred 30.12.2009]

The free book describes the experiences by Kniber. It is a useful guide for learning the basics of Scrum implementation and the combination of Scrum and Extreme Programming.

Kniber Henrik, 2008, 10 Ways to Screw up with Scrum and XP, Presentation at Agile 2008 Toronto, available at http://www.crisp.se/henrik.kniberg/presentations/agile2008/10-ways-to-screw-up-with-scrum-and-xp.pdf [referred 11.1.2010]

Useful tips how to avoid and identify the common pitfalls of implementing Scrum are introduced by the article.

Koponen Juho, 2008, Agile Release Planning in a Product Backlog Tool, Master's Thesis, Helsinki University of Technology, available at http://www.tml.tkk.fi/~anttiyj/Koponen-Agile.pdf [referred 11.1.2010]

The master's thesis identifies requirements for the product backlog management tool from the release planning perspective.

Koskela Lasse, 2009, Scrum: Ketterien menetelmien markkinajohtaja, Reaktor Innovations Oy, available at http://ttlry-fi-bin.directo.fi/@Bin/4ad365e2f208b8ab518129da61aaed16/1262871658/application/pdf/11062393/04_ScrumMarketLeaderOfAgileMethods_handout_LasseKoskela.pdf [referred 7.1.2010]

(In Finnish) The paper describes the history and the basics of Scrum. The paper includes examples of the backlogs.

Kähkönen Tuomo, 2004, Agile Methods for Large Organizations - Building Communities of Practice, Proceedings of the Agile Development Conference 2004, 22-26.6.2004, p. 2-10, IEEE ISBN 0-7695-2248-3, DOI 10.1109/ADEVC.2004.4

The paper presents three agile methods at Nokia.

Laanti Maarit, 2008, Implementing Program Model with Agile Principles in a Large Software Development Organization, p. 1383-1391, IEEE ISSN 0730-3157, DOI 10.1109/COMPSAC.2008.116

The article discusses the implementation of the multi-level backlogs at Nokia.

Larman Craig & Vodde Bas, 2009, Scaling Lean & Agile Development, Addison-Wesley, USA, ISBN 978-0-321-48096-5

The book presents principles of scaling Scrum.

Law Amy & Charron Raylene, 2005, Effects of Agile Practices on Social Factors, Proceedings of the 2005 workshop on Human and social factors of software engineering, 16.5.2005, St. Louis, Missouri, USA, p. 1-5 ACM ISBN 1-59593-120-1/05/05

The article describes the changes that Scrum introduced from the social factors perspective.

Leffingwell Dean, 2007, Chapter 18 excerpt from Scaling Software Agility: Best Practices for Large Enterprises, Addison Wesley, 2007, available at http://scalingsoftwareagility.files.wordpress.com/2009/03/whitepaper_systems-of-systems-and-the-agile-release-train2.pdf [referred 8.2.2010]

The excerpt of the book provides insights to the release management and the synchronized sprints between the teams.

Leszak Marek & Meier Manfred, 2007, Successful Global Development of a Large-scale Embedded Telecommunications Product, Global Software Engineering, 27-30.8.2007, p. 23-32, Munich Germany, IEEE ISBN 978-0-7695-2920-2, DOI 10.1109/ICGSE.2007.41

The article describes key success factors of software development at Alcatel-Lucent.

Lindvall Mikael, 2004, Agile Software Development in Large Organizations, Computer, Vol. 37, Issue 12, p. 26-34, IEEE ISSN 0018-9162, DOI 10.1109/MC.2004.231

The article describes experiences of agile, especially Extreme Programming, in Motorola, Nokia, ABB, and DaimlerChrysler. The companies are part of Software Experience Center (SEC).

Lowery Mike & Evans Marcus, 2007, Scaling Product Ownership, Agile 2007, 3-17.8.2007, p. 328-333, IEEE ISBN 0-7695-2872-4, DOI 10.1109/AGILE.2007.51

The article discusses the problems and remedies at BBC while scaling Scrum.

Lyon Richard & Evans Marcus, 2008, Scaling Up - Pushing Scrum out of its Comfort Zone, Agile 2008 Conference, Toronto, p. 395-400, IEEE ISBN 978-0-7695-3321-6, DOI 10.1109/Agile.2008.19

The article describes the problems BBC faced while scaling up Scrum from one to nine teams. Multiple useful advices are stated.

Marchenko Artem & Abrahamsson Pekka, 2008, Scrum in a Multiproject Environment: An Ethnographically-Inspired Case Study on the Adoption Challenges, Agile 2008 Conference, p. 15-26, IEEE ISSN 978-0-7695-3321-6, DOI 10.1109/Agile.2008.77

The paper presents challenges of implementing Scrum at Nokia.

Microguru Corporation, 2008, Scrum Sprint Planning and Execution with Virtual SCRUM BOARD, available at http://virtualscrumboard.com/scrum_sprint_planning_and_execution_with_virtual scrumboard.pdf [referred 8.1.2010]

The paper describes the functionality of an application called Virtual Scrum Board. Also the backlog and the burndown charts are discussed.

Mills Harlan D. et al. 1987, Cleanroom Software Engineering, Software IEEE, September 1987, Vol. 4, Issue 5, p. 19-25, IEEE ISSN 0740-7459, DOI 10.1109/MS.1987.231413

The paper presents the experiences of the cleanroom model implementation at IBM.

Murauskaite Asta & Adomauskas Vaidas, 2008, Bottelenecks in Agile Software Development Identified Using Theory of Constraints (TOC) Principles, Master's Thesis, IT University of Göteborg, available at http://gupea.ub.gu.se/dspace/bitstream/2077/10457/1/gupea_2077_10457_1.pdf [referred 17.2.2010]

The thesis presents six potential bottlenecks of lean identified at Ericsson in Sweden.

Nahor Ronen Bar and Katzav Erez, 2009, Enterprise Complex Backlog Management, Presentation at Germany Scrum Gathering October 2009, available at http://www.scrumalliance.org/resources/1106 [referred 8.1.2010]

The presentation discusses the means of managing the product backlog with different viewpoints.

Nerur Sridhar et al., 2005, Challenges of Migrating to Agile Methodologies, Communications of the ACM, May 2005, Vol. 48, No. 5, p.73-78, ISSN 0001-0782

The paper introduces experiences of changes and challenges while implementing Scrum.

Paasivaara Maria et al., 2008, Using Scrum in a Globally Distributed Project: A Case Study, Software Process Improvement and Practice, 2008, Vol. 13, p. 527-544, Wiley InterScience, DOI 10.1002/spip.402

The paper describes challenges and presents best practices of distributed scrum projects.

Paasivaara Maria et al., 2009, Using Scrum in Distributed Agile Development: A Multiple Case Study, 2009 Fourth IEEE Conference on Global Software Engineering, p. 195-204, IEEE ISSN 978-0-7695-3710-08, DOI 10.1109/ICGSE.2009.27

The paper describes challenges and presents best practices of distributed scrum projects.

Poppendieck Mary, 2002, Principles of Lean Thinking, Poppendicek.LLC, available at http://www.poppendieck.com/papers/LeanThinking.pdf [referred 21.4.2010]

The paper describes the origins of the lean principles and maps the lean thinking to the context of software development.

Poppendieck Mary, 2007, Lean Software Development, Software Engineering, ICSE 2007 Companion, 29th International Conference, 20-26.5.2007, Minneapolis, p. 165-166, IEEE ISBN 0-7695-2892-9, DOI 10.1109/ICSECOMPANION.2007.46

This short paper describes the contents of a lean course.

Rally Software, 2005, A CIO's Playbook for Adopting the Scrum Method of Achieving Software Agility with Dean Leffingwell and Hubert Smits, available at

http://www.rallydev.com/documents/CIO_Playbook_For_Adopting_Scrum_080805.pdf [referred 17.2.2010]

The whitepaper shortly describes the Scrum framework and then presents a framework for implementing Scrum in steps.

Racheva Zornitza & Daneva Maya, 2008, Using Measurements to Support Real-Option Thinking in Agile Software Development, Proceedings of the 2008 international workshop on Scrutinizing agile practices or shoot-out at the agile corral, Leipzig Germany, p. 15-18, ACM ISBN 978-1-60558-021-0

The article describes means to implement real-options valuation method to the agile software development..

Raman Sowmyan, 1998, Lean Software Development - Is It Feasible?, Digital Avionics Systems Conference 1998 Proceedings, 17th DASC, 31.10-7.11.1998, Bellevue, Vol. 1, p. C13/1 - C13/8, ISBN 0-7803-5086-3, DOI 10.1109/DASC.1998.741480

The article presents lean principles in the embedded software development. The author is working at Boeing.

Rawsthorne Dan, 2008, Complex Backlogs, Presentation at Chigago Scrum Gathering 2008, available at http://www.scrumalliance.org/resources/340 [referred 8.1.2010]

The presentation provides insights to large backlogs in case of multi-team Scrum.

Rendell Andrew, 2008, Effective and Pragmatic Test Driven Development, Agile 2008 Conference, p. 298-303, IEEE ISSN 978-0-7695-3321-6, DOI 10.1109/Agile.2008.45

The article describes the enhancements in code post implementing Test Driven Development at T-Mobile in the United Kingdom.

Royce Winston W., 1970, Managing the development of large software systems, Proceedings, IEEE Wescon August 1970, p. 1-9

In this paper Royce introduced the Waterfall model for the first time.

Santamaria Marina Gil, 2007, Agile & Scrum: What are these methodologies and how will they impact QA/testing roles?, Oracle White Paper, available at http://www.oracle.com/technology/tech/qa-testing/pdf/agile-scrum-qa-perspective.pdf [referred 5.12.2009]

The white paper describes the top-three enhancements Scrum introduced at Oracle.

Schilling Melissa A., 2004, Strategic Management of Technological Innovation, first edition, McGraw-Hill/Irwin, ISBN 978-0072942989

The book discusses innovation processes and management of the new product development.

Schwaber Ken, 1996, SCRUM Development Process, OOPSLA 1995 seminar, available at http://jeffsutherland.com/oopsla/schwapub.pdf [referred 30.12.2009]

The first paper introducing the Scrum framework (based on the seminar presentation by Schwaber and Sutherland).

Schwaber Ken, 2007, What is Scrum?, Handouts 16.8.2007, CSC resources, available at http://www.scrumalliance.org/resource_download/227 [referred 7.1.2010]

The paper describes the Scrum framework and provides valuable insights to the product backlogs.

Schwaber Ken, 2008, It's Not Scrum If..., Presentation at Stockholm Scrum Gathering Fall 2008, available at http://www.scrumalliance.org/resource_download/441 [referred 7.1.2010]

The presentation describes pitfalls of Scrum, focusing on the product backlog, the burndown chart, the sprint review, and the velocity.

Schwaber Ken, 2009, Scrum Guide, Online Guide, available at http://www.scrumalliance.org/resource_download/598 [referred 7.1.2010]

The paper is a brief guide to Scrum written by the co-founder of the Scrum framework and the Scrum alliance.

Shuping Liu & Ling Pang, 2008, The research of V model in testing embedded software, International Conference on Computer Science and Information Technology 2008, IEEE ISSN 978-0-7695-3308-7, DOI 10.1109/ICCSIT.2008.51

The article identifies and provides remedies to problems of the v-model.

Silverman David, 2006, Interpreting Qualitative Data, third edition, SAGE Publications Ltd, London, United Kingdom

The book describes alternative methods for gathering and analyzing qualitative data.

Stuart Jenny, 2009, 10 Keys to Successful Scrum Adoption, version 1.1, Construx white paper, available at http://www.construx.com/File.ashx?cid=2917 [referred 25.1.2010]

The paper introduces the basics of the Scrum framework and provides suggestions for the implementation of Scrum.

Sulaiman Tamara et al., 2006, AgileEVM - Earned Value Management in Scrum Projects, Agile Conference 2006, 23-28.7.2006, Minneapolis USA, IEEE ISBN 0-7695-2562-8, DOI 10.1109/AGILE.2006.15

The article introduces a mathematical model to qualitative analysis of Scrum. Also the needs of the different stakeholders are discussed.

Sutherland Jeff, 2001, Agile Can Scale: Inventing and Reinventing SCRUM in Five Companies, Cutter IT Journal Vol. 14, No. 12 2001, available at http://jeffsutherland.com/papers/scrum/Sutherland2001AgileCanScaleCutter.pdf [referred 5.1.2010]

The paper introduces findings from five different companies including discussion regarding hyper-productivity via integrating Scrum and Extreme Programming.

Sutherland Jeff, 2004, Agile Development: Lessons Learned from the First Scrum, available at http://jeffsutherland.com/scrum/FirstScrum2004.pdf [referred 31.12.2009]

The paper presents the findings from the very first Scrum.

Sutherland Jeff, 2005, Future of Scrum: Parallel Pipelining of Sprints in Complex Projects, AGILE 2005 Conference Denver, http://jeffsutherland.com/scrum/SutherlandFutureOfScrumAgile2005.pdf [referred 14.1.2010]

The paper describes three different types of Scrum in a large-scale R&D.

Sutherland Jeff, 2008, Pretty Good Scrum: Secret Sauce for Distributed Teams, Presentation on 17.12.2008, available at http://confluence.agilefinland.com/download/attachments/884822/Pretty+Good+Scrum+v6+CSM.pdf?version=1 [referred 11.1.2010]

The presentation introduces the scoring system for Scrum implementation (based on Nokia Networks case by Bas Vodde, 2006) and defines common pitfalls of the Scrum implementation.

Sutherland Jeff, 2009, Practical Roadmap to Great Scrum, Presentation at Germany Scrum Gathering 2009, available at http://www.scrumalliance.org/resources/1116 [referred 8.1.2010]

The presentation describes key success factors in the Scrum implementation. Also concepts that most companies mistakenly discard are discussed.

Sutherland Jeff et al., 2009, Shock Therapy: A Bootstrap for Hyper-Productive Scrum, available at

http://jeffsutherland.com/scrum/SutherlandShockTherapyAgile2009.pdf [referred 5.1.2010]

The paper describes the means of gaining hyper-productivity with Scrum. Some pitfalls are identified. Also suggestions for coaching new scrum teams are offered.

Sutherland Jeff & Altman Igor, 2010, Organizational Transformation with Scrum: How a Venture Capital Group Gets Twice as Much Done with Half the Work, 43rd Hawaii International Conference on Software Systems, Kauai, Hawaii 2010, available at http://jeffsutherland.com/scrum/SutherlandTakeNoPrisonersHICSS2010.pdf [referred 11.1.2010]

The paper introduces a case study at OpenView. The experiences from the first trial to scaling are introduced.

Takeuchi Hirotaka & Nonaka Ikujiro, 1986, The New New Product Development Game, Harvard Business Review January-February 1986, available at http://apln-richmond.pbworks.com/f/New+New+Prod+Devel+Game.pdf [referred 31.12.2009]

This paper was the first one to involve the term 'Scrum'. The paper studies R&D in multiple industries.

Tengshe Ash & Noble Scott, 2007, Establishing the agile PMO: managing variability across projects and portfolios, Agile 2007, 13-17.8.2007, Washington DC, USA, p. 188-193, IEEE ISBN 0-7695-2872-4, DOI 10.1109/AGILE.2007.24

The key findings from the article are options for metrics in Scrum and the lessons of scaling the portfolio management.

The French Scrum User Group, 2009, A National Survey on Agile Methods in France, The French Scrum User Group, June 2009, available at http://www.frenchsug.org/download/attachments/591296/National_survey_FrenchSUG_ENGL_en.pdf?version=2 [referred 15.2.2010]

The study presents the findings of a survey of 150 French companies using agile software development methods. Also telecommunication domain companies are involved in the study.

Wahl Andrew, 2009, NORTEL R.I.P.?, Canadian Business, 21.7.2009, Vol. 82 Issue 12/13, p. 40-41

The article discusses the fall of Nortel.

Woi Hin Kee, 2006, Future Implementation and Integration of Agile Methods in Software Development and Testing, Innovations in Information Technology, Nov 2006, Dubai, p. 1-5, IEEE ISBN 1-4244-0674-9, DOI 10.1109/INNOVATIONS.2006.301945

The paper provides insights to integrating Scrum and traditional software development methods. The paper relies on findings at Motorola.

Vodde Bas, 2006, Nokia Networks and Agile Development, presentation on 29.8.2006, available at hhttp://jeffsutherland.com/scrum/BasVodde2006_nokia_agile.pdf [referred 11.1.2010]

The presentation describes the process of implementing agile at Nokia Networks in 2006. This case study is the basis for Scrum implementation scoring system called the Nokia test.

Vähäniitty Jarno, 2005, A Tentative Framework for Connecting Long-Term Business and Product Planning with Iterative & Incremental Software Product Development, Proceedings of the seventh international workshop on Economics-driven software engineering research, 15.5.2005, St. Louis, Missouri USA, p. 1-4, ACM ISBN 1-59593-118-X

The paper presents a framework for managing different levels of planning, i.e., from iteration planning to the strategic release management.

Appendix A – The Agile Manifesto Principles

In 2001, a group of 17 software development experts decided twelve principles that formed the agile manifesto. The principles are as follows:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4. Business people and developers work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7. Working software is the primary measure of progress.

8. Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.

10. Simplicity—the art of maximizing the amount of work not done—is essential.

11. The best architectures, requirements and designs emerge from self-organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

(Fowler & Highsmith, 2001)

Appendix B – The Nokia Test

Bas Vodde invented the Nokia test for evaluating the feasibility of the Scrum implementation of a company. Jeff Sutherland enhanced the test. By answering to eight multiple-choice questions, an organization can avoid pitfalls of the Scrum implementation. Table B.1 summarizes the test. (Sutherland, 2008)

Table B.1. The Nokia test (Sutherland, 2008)

Criteria	Points
Question 1 - The Iterations	
No iterations	0
Iterations > 6 weeks	1
Iterations with variable length, < 6 weeks	2
Fixed iteration length, 6 weeks	3
Fixed iteration length, 5 weeks	4
Fixed iteration length, 4 weeks or less	10
Question 2 - The Testing	
No dedicated quality assurance	0
Unit tested	1
Feature tested	5
Features tested as soon as completed	7
Software passes acceptance testing	8
Software is deployed	10
Question 3 - The Agile Specification	
No requirements	0
Big requirements documents	1
Poor user stories	4
Good requirements	5
Good user stories	7
Just enough, just-in-time specifications	8
Good user stories tied to specifications as needed	10
Question 4 - The Product Owner	
No product owner	0
Product owner who does not understand Scrum	1
Product owner who disrupts team	2
Product owner not involved with team	2
Product owner with clear product backlog estimated by team before sprint planning meeting (READY)	5
Product owner with release roadmap with dates based on team velocity	8
Product owner who motivates team	10
Question 5 - The Product Backlog	
No product backlog	0
Multiple product backlogs	1
Single product backlog	3
Product backlog clearly specified and prioritized by ROI before sprint planning (READY)	5
Product owner has release plan based on product backlog	7
Product owner can measure ROI based on real revenue, cost per story point, or other metrics	10

Question 6 - The Estimates	
Product backlog is not estimated	0
Estimates not produced by team	1
Estimates not produced by planning poker	5
Estimates produced by planning poker by team	8
Estimate error < 10%	10
Question 7 - The Burndown Chart	
No burndown chart	0
Burndown chart not updated by team	1
Burndown chart in hours/days not accounting for work in progress (partial tasks burndown)	2
Burndown chart only burns down when task in done	4
Burndown only burns down when story is done	5
Add three points if team knows velocity	
Add two points if product owner bases release plan on known velocity	
Question 8 - The Team Disruption	
Manager or project leader disrupts team	0
Product owner disrupts team	1
Managers, project leaders or team leaders assigning tasks	3
Have project leader and scrum roles	5
No one disrupting team, only scrum roles	10

Appendix C – The Matrix for Interview Framework

In the semi-structured interviews, a blank version of the matrix combining the decision points and the stakeholders was utilized. Table C.1 illustrates the blank matrix.

Table C.1. The matrix for semi-structured interviews

	F0 One-pager request		F1 PO, PB		F2 Investment Decision		F3 Commitment		F4 Done
Innovation Process									
Early-Phase Program (EPP)									
Cross-functional Scrum Teams (xFT)									
Scrum Masters (SM)									
Product Owner Team (PO and PPOs)									
Release verification									
Product release management and post-GA activities									

Appendix D – The Stakeholder Matrix

Table D.1 summarizes the findings from interviews mapped to literature. Similarities are marked with 'positive', whereas the differences are marked with 'negative'.

Table D.1. The stakeholders and decision points

		The Innovation Process		The Early Phase Program (EPP)	
		Current thinking	**Reflection to literature**	**Current thinking**	**Reflection to literature**
	Early investigation	- In future: input to the EPP team from the innovation backlog	- Negative: Scrum framework includes built-in innovation management since items can be managed in PB (Barton, 2009) - Negative: ideas can be placed in the PB (Racheva & Daeva, 2008) - Negative: only one PB suggested (Laanti, 2008; Hornos & Izquiero, 2009; Racheva & Daneva, 2008; Sutherland, 2008)	- Input from customers through the product management - In future: Input from the innovation process	- Positive: change request are accepted by the agile manifesto (Fowler & Highsmith, 2001) - Positive: the EPP is similar to opportunity team concept (Nahor & Katzav, 2009)
F0	**One-pager request**	- No contribution		- Based on input, identify the need for one-pager	- Positive: lightweight documents (Kähkönen, 2004; Sutherland, 2008)
	Conduct a one-pager			- Early investigation leading to a one-pager - The one-pager is placed in a separated list - Uncertain whether the one-pagers are to be mapped to the product portfolio	- Positive: lightweight documents (Kähkönen, 2004; Sutherland, 2008) - Negative: ideas can be placed in the PB (Racheva & Daeva, 2008) - Negative: different stakeholders are to be involved early (Kähkönen, 2004) - Negative: avoid the silos of knowledge (Kalliney, 2009) - Negative: only one PB suggested (Laanti, 2008; Hornos & Izquiero, 2009; Racheva & Daneva, 2008; Sutherland, 2008)
F1	**Feature to the PB**			- Handover to the PO team that places the item in the PB	- Negative: handovers violate lean (Benefield, 2008; Poppendieck, 2002) - Negative: knowledge gaps may emerge (Kettunen, 2009; Woi Hin, 2006)
	FCS	- Enhancements to the new feature in the limits of the one-pager - Feedback loop to the EPP	- Negative: feedback loops can generate delays that violate lean (Benefield, 2008)	- No contribution	
F2	**Investment decision**	- No contribution			
	Implement	- Enhancements to the new feature in the limits of the one-pager - Feedback loop to the EPP - Suggestion: dedicated time for the scrum teams to complete items from the innovation backlog	- Negative: Scrum framework includes built-in innovation management since items can be managed in PB (Barton, 2009) - Negative: only one PB suggested (Laanti, 2008; Hornos & Izquiero, 2009; Racheva & Daneva, 2008; Sutherland, 2008) - Positive: creativity is enhanced by the SM (Eskelinen et al., 2010)		
F3	**Commitment**	- No contribution			
	Implement	- Enhancements to the new feature in the limits of the one-pager - Feedback loop to the EPP - Suggestion: dedicated time for the scrum teams to complete items from the innovation backlog	- Negative: Scrum framework includes built-in innovation management since items can be managed in PB (Barton, 2009) - Positive: creativity is enhanced by the SM (Eskelinen et al., 2010) - Negative: only one PB suggested (Laanti, 2008; Hornos & Izquiero, 2009; Racheva & Daneva, 2008; Sutherland, 2008)		
F4	**Done**	- No contribution			

Table D.1. The stakeholders and decision points (continued)

		The Cross-functional Scrum Teams (xFT)	
		Current thinking	**Reflection to literature**
	Early investigation	- No contribution	- Negative: knowledge gaps may emerge (Kettunen, 2009; Woi Hin, 2006) - Negative: different stakeholders are to be involved early (Kähkönen, 2004) - Negative: avoid the silos of knowledge (Kalliney, 2009)
F0	**One-pager request**		
	Conduct a one-pager		
F1	**Feature to the PB**	- No contribution - Suggestion: organize user story workshops	- Negative: knowledge gaps may emerge (Kettunen, 2009; Woi Hin, 2006) - Negative: avoid the silos of knowledge (Kalliney, 2009) - Negative: different stakeholders are to be involved early (Kähkönen, 2004) - Positive (suggestion): splitting the items by the scrum team (Auvinen et al., 2008)
	FCS	- No contribution - In future: possibly contribute to the feature concept study	- Negative: knowledge gaps may emerge (Kettunen, 2009; Woi Hin, 2006) - Negative: avoid the silos of knowledge (Kalliney, 2009) - Negative: different stakeholders are to be involved early (Kähkönen, 2004) - Positive: (suggestion): different stakeholders are to be involved early (Kähkönen, 2004) - Positive: (suggestion): elaborate lightweight documents in cross-functional workshops (Kähkönen, 2004)
F2	**Investment decision**	- No contribution to the decision - A partial handover from the PO team - Suggestion: organize user story workshops	- Negative: knowledge gaps may emerge (Kettunen, 2009; Woi Hin, 2006) - Negative: avoid the silos of knowledge (Kalliney, 2009) - Negative: handovers violate lean (Benefield, 2008; Poppendieck, 2002) - Positive (suggestion): splitting the items by the scrum team (Auvinen et al., 2008)
	Implement	- Select items form the PB in the sprint planning meeting - Attend the PB grooming session - Utilize the PB as a gateway to information - Sprint follow-up on a whiteboard	- Positive: only the scrum team selects the items (Schwaber, 2009) - Positive: the PB grooming sessions are held (Schwaber, 2009) - Positive: teams split items (Auvinen et al., 2008) - Positive: different sized items (Kniber, 2007) - Positive: the team needs to know the velocity (Koponen, 2008) - Positive: the PB includes links to a wiki (Deemer et al., 2008) - Positive: a physical whiteboard is needed for the sprint follow-up (Kniber, 2007)
F3	**Commitment**	- Commit to the feature release date and scope	- Positive: the date of feature completion is in the interests of the release management (Sulaiman et al., 2006)
	Implement	- Select items form the PB in the sprint planning meeting - Attend the PB grooming session - Utilize the PB as a gateway to information - Sprint follow-up on a whiteboard	- Positive: only the scrum team selects the items (Schwaber, 2009) - Positive: the PB grooming sessions are held (Schwaber, 2009) - Positive: teams split items (Auvinen et al., 2008) - Positive: different sized items (Kniber, 2007) - Positive: the team needs to know the velocity (Koponen, 2008) - Positive: the PB includes links to a wiki (Deemer et al., 2008) - Positive: a physical whiteboard is needed for the sprint follow-up (Kniber, 2007)
F4	**Done**	- The MMF is completed - Perform the handover to the release project n (optional: to the post-GA project n-1 and n-2)	- Positive: nothing extra added to the feature (Benefield, 2008) - Positive: internal release (Leffingwell, 2007) - Negative: the handovers violate lean (Benefield, 2008; Poppendieck, 2002)

Table D.1. The stakeholders and decision points (continued)

		The Scrum Masters (SM)		The Product Owner Team (PO and PPOs)	
		Current thinking	**Reflection to literature**	**Current thinking**	**Reflection to literature**
	Early investigation	- Mentor the EPP team for a few hours per week	- Positive: the SM mentors all stakeholders (Schwaber, 2009)	- No contribution	- Negative: knowledge gaps emerge (Kettunen, 2009; Woi Hin, 2006) - Negative: avoid the silos of knowledge (Kalliney, 2009) - Negative: different stakeholders are to be involved early (Kähkönen, 2004)
F0	**One-pager request**				
	Conduct a one-pager				
F1	**Feature to the PB**	- Mentor the EPP team for a few hours per week - In future: facilitate the possible user story writing sessions	- Positive: the SM mentors all stakeholders (Schwaber, 2009) - Positive: the SM facilitates the meetings (Eskelinen et al., 2010)	- Receive the responsibility of the feature - Place the feature as items into the PB	- Negative: handovers violate lean (Benefield, 2008; Poppendieck, 2002)
	FCS	- Mentor the PO team	- Positive: the SM mentors all stakeholders (Schwaber, 2009)	- Write the FCS - Create the user stories into the PB	- Positive: preliminary planning regarding the architecture (Kettunen, 2009) - Positive/Negative: lightweight documents (Kähkönen, 2004; Sutherland, 2008)
F2	**Investment decision**	- Mentor the partial handover - Facilitate the possible user story writing sessions in the future	- Positive: the SM mentors all stakeholders (Schwaber, 2009) - Positive: the SM facilitates the meetings (Eskelinen et al., 2010) - Negative: handovers violate lean (Benefield, 2008; Poppendieck, 2002)	- Present the feature to the product management that decides whether to invest or not - Perform the partial handover to the scrum teams	- Positive: the PO team is a gateway between the scrum teams and the business (Stuart, 2009) - Negative: handovers violate lean (Benefield, 2008; Poppendieck, 2002)
	Implement	- Mentor the scrum team - Facilitate the PB grooming sessions - Remove impediments	- Positive: the SM mentors all stakeholders (Schwaber, 2009) - Positive: the SM facilitates the meetings (Eskelinen et al., 2010) - Positive: Creativity is enhanced by the SM (Eskelinen et al., 2010) - Positive: the SM follow-up the velocity and the warning signs (Kniber, 2007)	- Calculate the velocity - Reprioritize the PB items - Split the PB items - Participate in the PB grooming sessions - Cost follow-up	- Positive: the PB is dynamic including all sized items created in the grooming sessions (Schwaber, 2009) - Positive: the PB is ready all the time (Deemer et al., 2008) - Positive: the PO is responsible for ROI (Eskelinen et al., 2010)
F3	**Commitment**	- Mentor the scrum teams and the PO team stating the commitment	- Positive: the SM mentors all stakeholders (Schwaber, 2009)	- Calculate the date and the scope for the release of the feature based on the velocity - Commit to the date and the scope	- Positive: the PO team is a gateway between the scrum teams and the business (Stuart, 2009) - Positive: the date of feature completion is in the interests of the release management (Sulaiman et al., 2006) - Positive: the PO is responsible for ROI (Eskelinen et al., 2010)
	Implement	- Mentor the scrum team - Facilitate the PB grooming sessions - Remove impediments	- Positive: the SM mentors all stakeholders (Schwaber, 2009) - Positive: the SM facilitates the meetings (Eskelinen et al., 2010) - Positive: Creativity is enhanced by the SM (Eskelinen et al., 2010) - Positive: the SM follow-up the velocity and the warning signs (Kniber, 2007)	- Calculate the velocity - Reprioritize the PB items - Split the PB items - Participate in the PB grooming sessions - Cost follow-up	- Positive: the PO is responsible for ROI (Eskelinen et al., 2010) - Positive: the PB is ready all the time (Deemer et al., 2008) - Positive: the PB is dynamic including all sized items created in the grooming sessions (Schwaber, 2009)
F4	**Done**	- The MMF is completed - Mentor the handover to the release project n (optional: to the post-GA n-1 and n-2)	- Positive: nothing extra added to the feature (Benefield, 2008) - Negative: handovers violate lean (Benefield, 2008; Poppendieck, 2002) - Positive: internal release (Leffingwell, 2007)	- Responsible for accepting the quality of the feature - Perform the handover to the release project n (optional: to the post-GA n-1 and n-2)	- Positive: nothing extra added to the feature (Benefield, 2008) - Negative: handovers violate lean (Benefield, 2008; Poppendieck, 2002) - Positive: internal release (Leffingwell, 2007)

Table D.1. The stakeholders and decision points (continued)

		The release verification		The product release management and the Post-GA activities	
		Current thinking	**Reflection to literature**	**Current thinking**	**Reflection to literature**
	Early investigation	- Support EPP only if requested	- Negative: avoid the silos of knowledge (Kalliney, 2009) - Negative: different stakeholders are to be involved early (Kähkönen, 2004)	- No contribution	- Negative: avoid the silos of knowledge (Kalliney, 2009) - Negative: different stakeholders are to be involved early (Kähkönen, 2004)
F0	**One-pager request**				
	Conduct a one-pager				
F1	**Feature to the PB**				
	FCS	- Participate in verification analysis - Identify dependencies	- Dependencies (hardware, external, internal) exists (Kettunen, 2009) - Positive: different stakeholders are to be involved early (Kähkönen, 2004)	- Prepare for the handover - Identify dependencies	- Dependencies (hardware, external, internal) exists (Kettunen, 2009)
F2	**Investment decision**	- No contribution	- Positive: different stakeholders are to be involved early (Kähkönen, 2004)	- Feature release date estimate known - Estimate for which product release the feature will be bundled	- Positive: the date of feature completion is in the interests of the release management (Sulaiman et al., 2006)
	Implement	- Support the scrum teams by executing requested test cases - Participate in the sprint demonstration meetings	- Positive: scrum-external system verification is needed (Kniber, 2007) - Positive: different stakeholders are to be involved early (Kähkönen, 2004)	- Prepare for the handover - Identify dependencies	- Dependencies (hardware, external, internal) exists (Kettunen, 2009) - Positive: different stakeholders are to be involved early (Kähkönen, 2004)
F3	**Commitment**	- The criteria for F4 should be known and communicated in addition to date, scope, and risks	- Positive: different stakeholders are to be involved early (Kähkönen, 2004)	- Feature release date and scope known, risks partly known - Evaluate the estimate for which product release the feature will be bundled	- Positive: the date of feature completion is in the interests of the release management (Sulaiman et al., 2006) - Positive: different stakeholders are to be involved early (Kähkönen, 2004)
	Implement	- Support the scrum teams by executing requested test cases - Participate in the sprint demonstration meetings	- Positive: scrum-external system verification is needed (Kniber, 2007) - Positive: avoid the silos of knowledge (Kalliney, 2009) - Positive: different stakeholders are to be involved early (Kähkönen, 2004)	- Prepare for the handover - Identify dependencies	- Dependencies (hardware, external, internal) exists (Kettunen, 2009) - Positive: avoid the silos of knowledge (Kalliney, 2009) - Positive: different stakeholders are to be involved early (Kähkönen, 2004)
F4	**Done**	- Product increment received from the scrum teams - The final system testing begins	- Negative: knowledge gaps emerge (Kettunen, 2009; Woi Hin, 2006) - Positive: scrum-external system verification is needed (Kniber, 2007) - Negative: handovers violate lean (Benefield, 2008; Poppendieck, 2002)	- Product increment received from the scrum teams - Evaluate the estimate for which product release the feature will be bundled	- Negative: handovers violate lean (Benefield, 2008; Poppendieck, 2002)